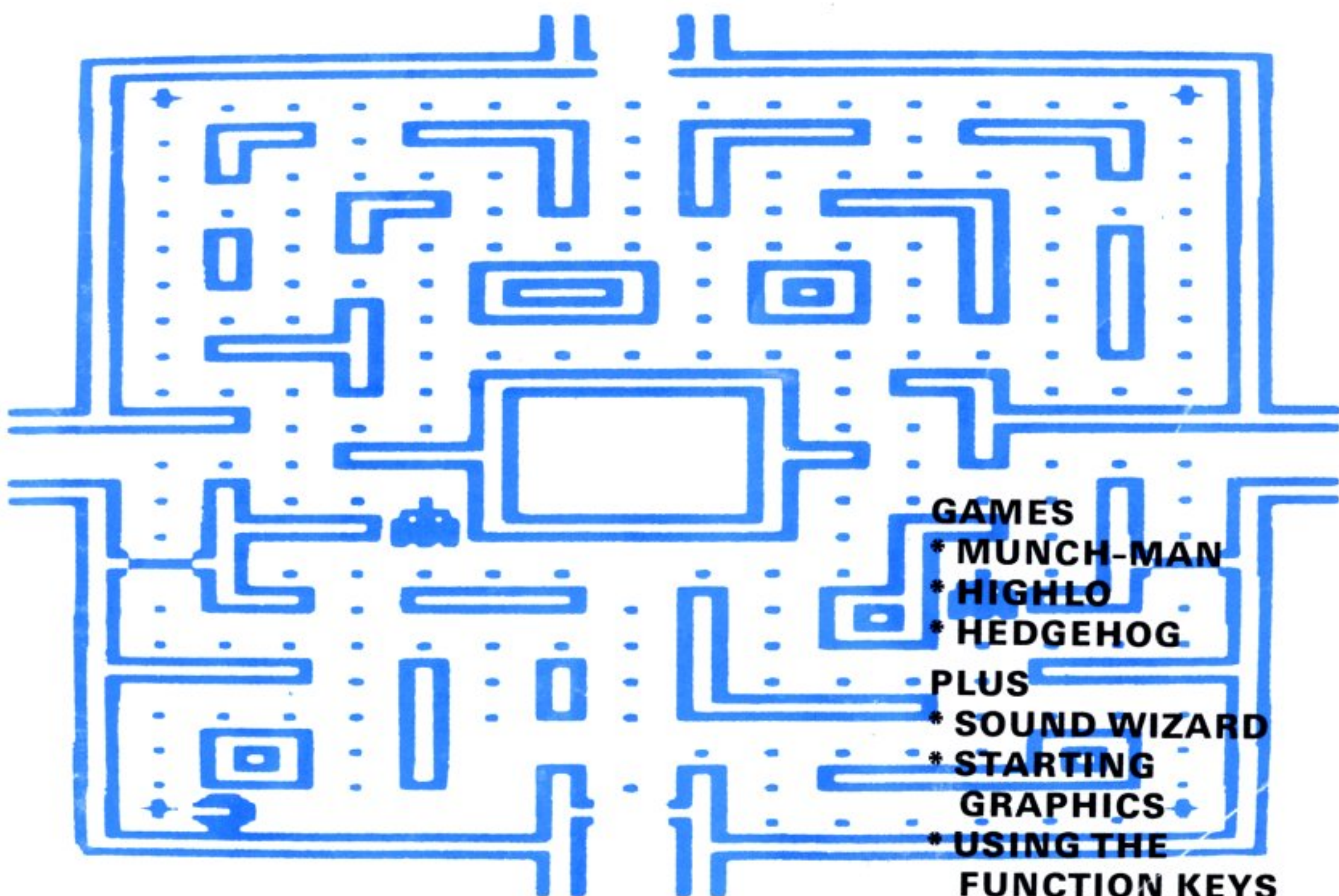


£1.00

# ELBUG

FOR THE **ELECTRON**

Vol 1 No 1 November 1983



## GAMES

- \* MUNCH-MAN
- \* HIGHLO
- \* HEDGEHOG

## PLUS

- \* SOUND WIZARD
- \* STARTING GRAPHICS
- \* USING THE FUNCTION KEYS

## PLUS

- \* NEWS, REVIEWS
- \* HINTS & TIPS

And much  
more



## EDITORIAL

Welcome to ELBUG. You will find in its pages a collection of programs, articles, hints and ideas that help you to get the most from your Electron. The Electron is a powerful machine, and it is our aim to support it to the full. If you have suggestions or contributions for the magazine, then please do not hesitate to contact us.

As you will see, we have called our magazine ELBUG, rather than ORBIT, the name of the user group. This is because the name ORBIT is already in use as a periodical title, and has been registered for that purpose.

We hope by now that you have been able to get your hands on an Electron, and have made good use of our Introductory Cassette. In spite of early indications, Acorn admit that Electrons are not coming onto the market in the volumes that they had intended. The demand for the machine is high, and it looks as if there will be a supply shortfall into the new year.

In this issue of ELBUG we have tried to present a balance of material. There are some introductory articles, software reviews, and games - Munch-man and Highlo, but also some utilities, including an extremely useful double height routine to enhance your own programs. The main program features in this issue are derived from past issues of BEEBUG, our sister magazine for the BBC micro. The programs have been selected because of their particular merit - Munch-man and Highlo are extremely good games, while Union Jack provides striking displays, and Double-Height and Sound Wizard are exceptionally useful utilities. In all cases the programs have been thoroughly tested on the Electron, and very often have been considerably modified to make them run smoothly and speedily on the Electron.

In future issues of ELBUG we shall introduce programs written especially for the Electron, though there will always be useful cross-fertilization of ideas between the two magazines. Incidentally, if you would like to purchase back issues of BEEBUG as a member of ORBIT, then you may do so at a BEEBUG member's price (see page 35).

As a member of ORBIT, you may also make full use of the discounts offered to BEEBUG members in the magazine supplement. This supplement at present has a strong BBC micro flavour, but we expect that advertisers will soon be introducing products specifically for the Electron.

BEEBUGSOFT, the software house of BEEBUG Publications Ltd. will be producing a range of software for the Electron. At present there is just one title available (further details on page 35). We shall shortly be expanding the range considerably; and next issue we will announce the first Software Club items for the Electron.

David Graham



# ELBUG MAGAZINE

## GENERAL CONTENTS

### PAGE CONTENTS

2	Editorial
4	Using Programs Published in Elbug
5	Munch-Man
8	The Acorn Electron Reviewed
11	Sound Wizard
13	Electron Graphics (Part 1)
16	Double Height Characters
17	Highlo Card Game
19	The First Electron Software from Acornsoft
22	Union Jack
23	Using the Programmable Function Keys
25	The Introductory Cassette
26	Three Tape Recorders for the Electron
28	News
29	Electron Book Index
31	Hedgehog

## HINTS & TIPS

### PAGE CONTENTS

10	RND(-TIME)
10	Finer Positioning of Text
10	Scanning Strings
12	SOUND Frequency
12	Graphics Origin
15	Break and Control Break
15	Silent Games
18	Logical/Physical Colours
21	Control Key Codes
21	Hex to Decimal and Back
34	TRACE
34	Self Validating GET Routine
34	Calculating X Squared
34	Program Length

## PROGRAMS

### PAGE CONTENTS

5	Munch-Man Game
11	Sound Wizard
13	Graphics Example Program
16	Double Height Characters
17	Highlo Card Game
22	Union Jack Display
23	Keyset Program
31	Hedgehog Game

## USING PROGRAMS PUBLISHED IN ELBUG

by Mike Williams

As this is the first issue of ELBUG, we thought it would be a good idea to give you some guidance about the programs we will be publishing in the magazine (including those in this issue), in order to help you get the best results from the printed listings.

All our programs are listed directly from computer in order to avoid any transcription or typographic errors. For those who wonder how this is possible with the Electron, the programs are developed and tested on that micro, before saving on cassette and then reloading on a BBC micro for printing.

Most of our ELBUG programs will be in Basic, but occasionally part of a program will be in a different language called assembler (or machine code). This is the language that your micro really uses and understands. Basic is provided in ROM in order to interpret programs in Basic in a way that can be understood by the Electron.

Another chip in your Electron contains the Operating System (O.S. for short). This is a special form of program which looks after the general organisation of what is going on inside your micro. It contains a set of very useful routines that, once in a while, we shall want to refer to directly.

When you are typing a program into your Electron from a listing in ELBUG, you must do this as accurately as you can. Even mistyping just a single character in a whole program can cause the program to malfunction. Spaces are important and should always be included whenever they appear in the listing. Normally spaces will be single spaces, but check with characters in the line above or below if you need to count how many spaces to enter. There is one exception here, the space immediately following the line number may be omitted. This is generated by the LISTO command for clarity.

Remember, as you type a program into your Electron, to terminate each line by pressing the return key. In the text of articles in ELBUG, where we want to indicate this, we represent it by <return> to make everything quite clear. However, to save space, we don't do this with complete listings of

programs. We will also use the angle brackets, < and >, similarly for other purposes when the need arises.

As you type in a program, it is a good idea to save a copy from time to time, even though the program is incomplete. This will avoid any catastrophe that might otherwise require you to retype everything. Do this particularly, if you want a rest or are going to leave the micro unattended for a while.

Once the outline program has been typed in and saved, you can try running the program. Don't be too dismayed if it doesn't work correctly first time. If you get an error message, then list the line referred to on the screen, and check carefully with the magazine listing. If you are sure that the line is correct, yet the computer persists in indicating an error in that line, then the problem is probably an error in an earlier line that has remained undetected until now. Note which variables are used in that line and check any (or all if necessary) of the previous lines in the program that also used those variables. This is a common answer, but you may have to think of other alternatives. When you have the program working correctly, it is a good idea to save two copies on cassette for added security.

We cannot, in a general guide like this, cope with all the possible errors you might generate. Sometimes there will be no other remedy than to check through the entire program from beginning to end checking every line. A careful and methodical approach to typing the program in to your Electron to start with can save an awful lot of time later. In particular be careful not to confuse the letter O with zero, or lower case L with one (these are printed differently in our listings); and be sure to keep spaces the way they are.





# MUNCH-MAN

by Andrew Hynes

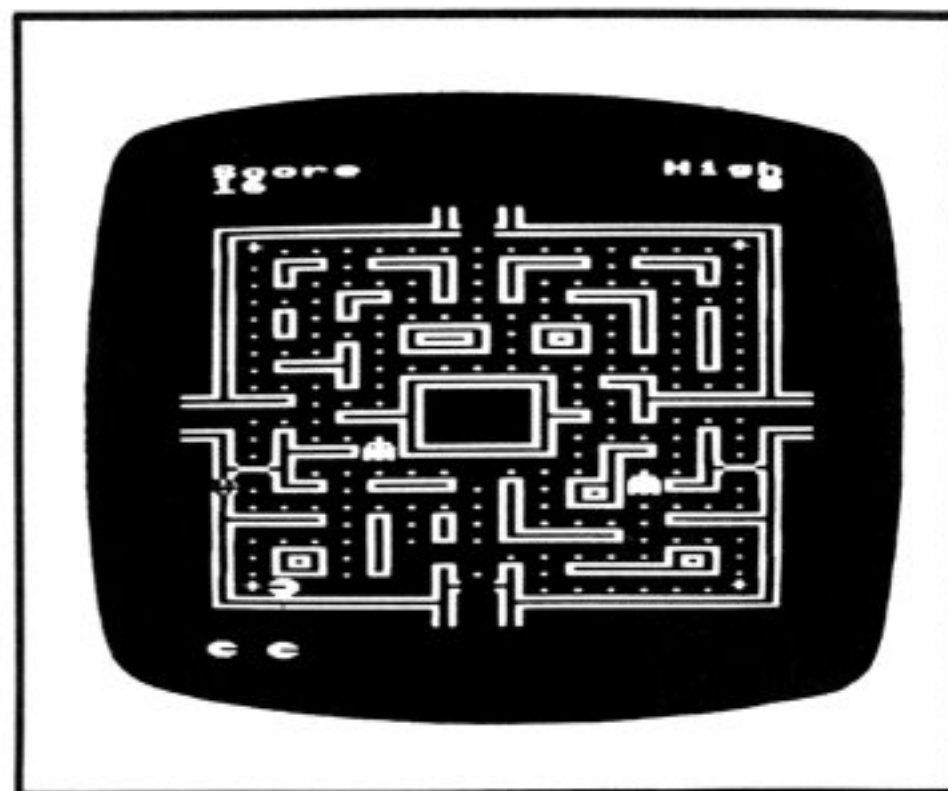


Munch-Man is a very well written version of the popular arcade game PAC-MAN. The game, although written in Basic, is quite fast and enjoyable. You take the part of a hungry 'mouth' scurrying around a maze, eating dots to score points. There are three ghosts that chase you and try to kill you. These ghosts move through walls, thus making it harder for you. To combat these ghosts you can eat a power pill, situated in the 4 corners of the maze, and these cause the ghosts to turn yellow and run away. While the ghosts are in this state you can eat them to gain massive bonus points.



Once you have cleared a display, a new screen appears, and you carry on until you lose three lives. Happy eating!

The best strategy for this game is to make use of the power pills, and the tunnels - because in this implementation the ghosts do not chase you through the tunnels.



```

200 PROCprintscore(0):PRINTTAB(19-LE
N(STR$(H%)),1);H%:COLOUR3:PRINTTAB(1,29
)STRING$(lives%-1,CHR$237+" ") " "
210 REPEAT:*FX 15,0
220 ghost=ghost+1:IF ghost=3:ghost=0
230 PROCpacman:PROCcheck
240 IF EAT=1:SOUND1,-14,150,1
250 IFTIME>2000ANDfruit=0ANDtot%<150:
PROCfruit
260 IFRND(1)>.9:PROCdoor
270 IFfruit=1ANDTIME>3000/sheet%:PRO
Cnofruit
280 IFTIME>1500/sheet% ANDEAT=1:EAT=0
290 UNTIL tot%=187 OR KILL=1
300 IF tot%=187:GOTO 180
310 PROCkilled:IF lives%>0:GOTO 200
320 PROCend:UNTIL FALSE
330:
340 DEFFNcalcX(X)
350 IF X>19:X=0
360 IF X<0:X=19
370 =X
380:
390 DEFFNcalcY(Y)
400 IF Y>24:Y=0
410 IF Y<0:Y=24
420 =Y
430:
440 DEFPROCtime(T)
450 TM=TIME:REPEAT UNTIL TIME>TM+T
460 ENDPROC
470:
480 DEFPROCprintscore(S%)
490 score%=score%+S%
500 COLOUR7:PRINTTAB(1,1);score%
510 ENDPROC
520:
530 DEFPROCpacman

```

```

10 REM Program Munch-Man
20 REM Author A.Hynes
30 REM Version El.0
40 REM ELBUG November 1983
50 REM Program subject to Copyright
60:
70 *FX 11,1
80 *FX 12,1
90 ON ERROR GOTO 2150
100 MODE5:VDU23,1,0;0;0;0;0;
110 PROCCHARSET
120 ENVELOPE 1,1,4,-4,4,10,20,10,126,
0,0,-126,126,126
130 H%=0:DIM maze$(19,24),ghost(2,2)
140 REPEAT
150 PRINTTAB(5,2)"MUNCHMAN"TAB(6,9)"*
.....UP"TAB(6,11)"?...DOWN"TAB(6,13)"Z.
..LEFT"TAB(6,15)"X..RIGHT"
160 VDU17,6,31,6,21,238,17,1,32,32,24
3,32,243,32,243:PROCspace
170 lives%=3:score%=0:sheet%=0:ghost=
0:dir$=" "
180 PROCsetupmaze:TIME=0:fruit=0:EAT
=0
190 COLOUR7:PRINTTAB(1,0)"Score"TAB(1
4)"High"

```



```

540 OX%=pacX%:OY%=pacY%:SC=score%:IF
pac$<>CHR$246:OP$=pac$
550 dir$=INKEY$(0)
560 IF dir$=":" pac$=CHR$239: pacY%=pa
cY%-1
570 IF dir$="/" pac$=CHR$240: pacY%=pa
cY%+1
580 IF dir$="Z" pac$=CHR$238: pacX%=pa
cX%-1
590 IF dir$="X" pac$=CHR$237: pacX%=pa
cX%+1
600 pacY%=FNcalcY(pacY%): pacX%=FNcalc
X(pacX%)
610 IF dir$="" AND pac$=CHR$246 THEN
pac$=OP$ ELSE IF dir$="" pac$=CHR$246
620 A$=maze$(pacX%,pacY%)
630 IF A$="":PROCprintscore(2):SOUN
D1,1,60,1:tot%=tot%+1
640 IF A$=CHR$244:PROCprintscore(100
):PROCnofruit:SOUND1,1,150,2
650 IF A$="+":PROCprintscore(50):EAT
=1:TIME=0:SOUND1,1,100,2
660 IF SC=score% AND A$<>" ":pacX%=OX
%:pacY%=OY%
670 VDU17,3,31,OX%,OY%+3,32
680 PRINTTAB(pacX%,pacY%+3);pac$
690 maze$(pacX%,pacY%)=" "
700 ENDPROC
710:
720 DEFPROCruboutghost(G%):COLOUR1
730 M$=maze$(ghost(G%,1),ghost(G%,2))
740 IF M$="." OR M$="+":COLOUR2
750 IF M$=CHR$244:COLOUR1
760 IF M$=CHR$245:COLOUR5
770 PRINTTAB(ghost(G%,1),ghost(G%,2)+
3)maze$(ghost(G%,1),ghost(G%,2))
780 ENDPROC
790:
800 DEFPROCghost(G%):C=0
810 PROCruboutghost(G%)
820 IF EAT=1:COLOUR2 ELSE COLOUR1
830 IFEAT=0:DX%=1:DY%=1 ELSE DX%=-1:D
Y%=-1
840 IFG%=0:PROCvert:IFC=0:PROCchoriz
850 IFG%=1:PROCchoriz:IFC=0:PROCvert
860 IFG%=2:PROCchoriz:PROCvert
870 PRINTTAB(ghost(G%,1),ghost(G%,2)+
3)CHR$243
880 ENDPROC
890:
900 DEFPROCvert
910 A=FNcalcY(ghost(G%,2)+DY%)
920 B=FNcalcY(ghost(G%,2)-DY%)
930 IFpacY%>ghost(G%,2):ghost(G%,2)=A
:C=1
940 IFpacY%<ghost(G%,2):ghost(G%,2)=B
:C=1
950 ENDPROC
960:
970 DEFPROCchoriz
980 A=FNcalcX(ghost(G%,1)+DX%)
990 B=FNcalcX(ghost(G%,1)-DX%)
1000 IFpacX%>ghost(G%,1):ghost(G%,1)=A
:C=1
1010 IFpacX%<ghost(G%,1):ghost(G%,1)=B
:C=1
1020 ENDPROC
1030:
1040 DEFPROCdoor
1050 D%=RND(4):RESTORE1880
1060 FORF%=1 TO D%:READX,Y:NEXT
1070 IF X=pacX% AND Y=pacY%:GOTO 1050
1080 IF maze$(X,Y)=" ":maze$(X,Y)=CHR$
245 ELSE maze$(X,Y)=" "
1090 VDU17,5,31,X,Y+3,ASC(maze$(X,Y))
1100 ENDPROC
1110:
1120 DEFPROCfruit
1130 VDU17,1,31,9,18,244
1140 fruit=1:TIME=0
1150 maze$(9,15)=CHR$244
1160 ENDPROC
1170:
1180 DEFPROCnofruit
1190 VDU31,9,18,32:fruit=0
1200 maze$(9,15)=" "
1210 ENDPROC
1220:
1230 DEFPROCcheck:KILL=0
1240 FOR G%=0 TO 2
1250 IF G%=ghost:PROCghost(G%)
1260 IF ghost(G%,1)<>pacX% OR ghost(G%
,2)<>pacY%:GOTO 1280
1270 IF EAT=1:PROCprintscore(250):SOU
ND1,1,250,1:PROCsetupghosts(G%) ELSE KI
LL=1
1280 NEXT
1290 ENDPROC
1300:
1310 DEFPROCKilled:COLOUR3
1320 FOR C=237 TO 242
1330 VDU31,pacX%,pacY%+3,C
1340 SOUND1,1,C/2,1:PROctime(20)
1350 NEXT
1360 VDU31,pacX%,pacY%+3,32
1370 FOR G%=0 TO 2
1380 PROCruboutghost(G%)
1390 PROCsetupghosts(G%)
1400 NEXT
1410 PROCpac:lives%=lives%-1
1420 ENDPROC
1430:
1440 DEFPROCend
1450 IF score%>H%:H%=score%
1460 COLOUR7:PRINTTAB(19-LEN(STR$(H%))
,1);H%
1470 PROCspace
1480 ENDPROC
1490:
1500 DEFPROCspace:*FX 15,0

```





```

1510 VDU17,15,31,1,30:PRINT"SPACE BAR
TO START"
1520 REPEAT UNTIL GET=32:VDU12,17,7
1530 ENDPROC
1540:
1550 DEFPROCsetupghosts(G%)
1560 RESTORE 1870
1570 FOR D%=0 TO G%:READ COL,X:NEXT
1580 ghost(G%,1)=X
1590 ghost(G%,2)=12
1600 ghost(G%,0)=COL
1610 ENDPROC
1620:
1630 DEFPROCpac
1640 pacX%=9:pacY%=20:pac$=CHR$237
1650 ENDPROC
1660:
1670 DEFPROCsetupmaze
1680  FORG%=0TO2:PROCsetupghosts(G%):N
EXT
1690 PROCpac:tot%=0:sheet%=sheet%+1
1700 COLOUR128:RESTORE 1890
1710 FOR D%=3 TO 27
1720 READ maze$
1730 FOR E%=0 TO 19
1740 A$=MID$(maze$,E%+1,1)
1750 IF A$>="A" AND A$<="M":M$=CHR$(AS
C(A$)+159):COLOUR1 ELSE M$=A$:COLOUR2
1760 PRINTTAB(E%,D%)M$
1770 maze$(E%,D%-3)=M$
1780 NEXT
1790 NEXT
1800 VDU19,2,9,0,0,0,19,4,14,0,0,0
1810 FOR D%=1 TO 21
1820 SOUND 2,-15,D%*2,1
1830 SOUND 1,-15,D%*3,1
1840 NEXT:VDU20
1850 ENDPROC
1860:
1870 DATA 1,1,5,9,6,18
1880 DATA 9,1,9,22,2,15,17,15
1890 DATA "      A A      "
1900 DATA" CBBBBBBL LBBBBBBBD "
1910 DATA" A+.....+A "
1920 DATA" A.CI.JBD.CBI.JBD.A "
1930 DATA" A.G....A.A....G.A "
1940 DATA" A...CI.G.G.JBD...A "
1950 DATA" A.H.G.....A.H.A "
1960 DATA" A.G...CBD.CD.A.A.A "
1970 DATA" A...H.EBF.EF.G.A.A "
1980 DATA" A.JBK.....G.A "
1990 DATA" A...G.CBBBD.JD...A "
2000 DATA"BLBI...A  A..MBBBLB"
2010 DATA" ...JBK  MI.G... "
2020 DATA"BD.H...A  A....H.CB"
2030 DATA" A.MBI.EBBBF.CI.A.A "
2040 DATA" M K.....A..M K "
2050 DATA" A.EI.JBI.H.CK.JF.A "

```

```

2060 DATA" A.....A.EF....A "
2070 DATA" MBI.H.H.A....JBBK "
2080 DATA" A....A.G.EBBI....A "
2090 DATA" A.CD.A.. ....CD.A "
2100 DATA" A.EF.G.H.H.JBBLF.A "
2110 DATA" A+.....M K.....+A "
2120 DATA" EBBBBBBK MBBBBBBBF "
2130 DATA"      A A      "
2140:
2150 ON ERROR OFF
2160 MODE 7: *FX12
2170 *FX15
2180 REPORT:PRINT" at line ";ERL
2190 END
2200:
2210 DEFPROCCHARSET
2220 VDU23,224,102,102,102,102,102,102
,102,102
2230 VDU23,225,0,255,255,0,0,255,255,0
2240 VDU23,226,0,127,127,96,96,103,103
,102
2250 VDU23,227,0,254,254,6,6,230,230,1
02
2260 VDU23,245,0,0,0,255,255,0,0,0
2270 VDU23,246,60,126,255,255,255,255,
126,60
2280 VDU23,228,102,103,103,96,96,127,1
27,0
2290 VDU23,229,102,230,230,6,6,254,254
,0
2300 VDU23,230,102,102,102,102,102,126
,126,0
2310 VDU23,231,0,126,126,102,102,102,1
02,102
2320 VDU23,232,0,254,254,6,6,254,254,0
2330 VDU23,233,0,127,127,96,96,127,127
,0
2340 VDU23,234,102,230,230,6,6,230,230
,102
2350 VDU23,235,102,231,231,0,0,255,255
,0
2360 VDU23,236,102,103,103,96,96,103,1
03,102
2370 VDU23,237,60,126,255,224,224,255,
126,60
2380 VDU23,238,60,126,255,7,7,255,126,
60
2390 VDU23,239,36,102,231,231,231,255,
126,60
2400 VDU23,240,60,126,255,231,231,231,
102,36
2410 VDU23,241,0,0,42,0,34,0,42,0
2420 VDU23,242,8,73,42,0,99,0,42,73
2430 VDU23,243,126,90,219,255,213,171,
255,219
2440 VDU23,244,198,56,108,222,190,222,
108,56
2450 ENDPROC

```

This program was adapted from a version for the BBC micro first published in Beebug Vol.2 No.5.



# THE ACORN ELECTRON REVIEWED

by David Graham and Mike Williams

The Electron is the third microcomputer to be produced by Acorn in almost as many years, and follows the Atom and the enormously successful BBC micro. Indeed, the Electron is not a completely new machine for it contains many of the features that have made the BBC micro so popular. In appearance, the Electron is quite new and its smart looks are sure to attract many customers. The price of £199 will also bring the Electron within the pocket of many potential purchasers. So what do Acorn provide for the money?



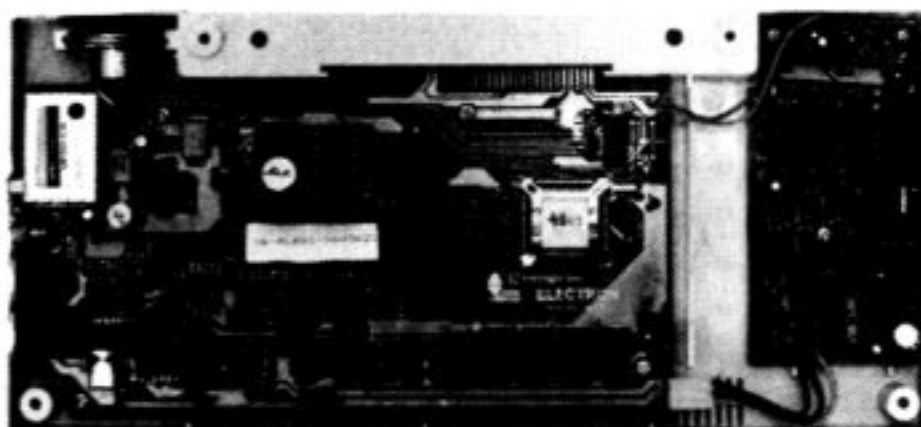
First of all, it looks as though the supply difficulties encountered by early BBC users will not be repeated with the Electron. Our review machine was a production model, and this is a good sign, as it suggests that production lines have been gearing up since well before the launch. In other ways too, Acorn have learned from their experiences with the Beeb. The packaging has been well thought through, and the documentation is far better than that received by early Beeb owners.

## ELECTRON PACK

Apart from the machine itself, the pack contains a large moulded plug with built-in transformer, a UHF TV lead, an introductory cassette, and two books. One of these is a manual, spiral bound for easy opening. This is well planned, with a detailed list of contents. A strong minus point is its lack of index, and we would urge Acorn to append an index to the release version of the manual. Apart from this the manual is excellent. The introductory chapters have clearly been written with great care to introduce the new owner

gently to the use of their new Electron. Later chapters provide a comprehensive guide to the Basic language and also to 6502 assembler.

The second book 'Start Programming with the Electron' by Masoud Yazdani (published by Addison-Wesley) is an admirable introductory text. What is so impressive about it, is that it introduces fully structured programming right from the start. In the first chapter it introduces the PRINT statement alongside PROC and ENDPROC. Procedures are fundamental to structured Basic, and it is nice to see them being introduced in this way. Our one serious reservation about this book is again its lack of index in the release version. We have, however, managed to obtain a full index from the author for publication in this first issue of ELBUG, and we understand that this index will be included in future editions of the book.

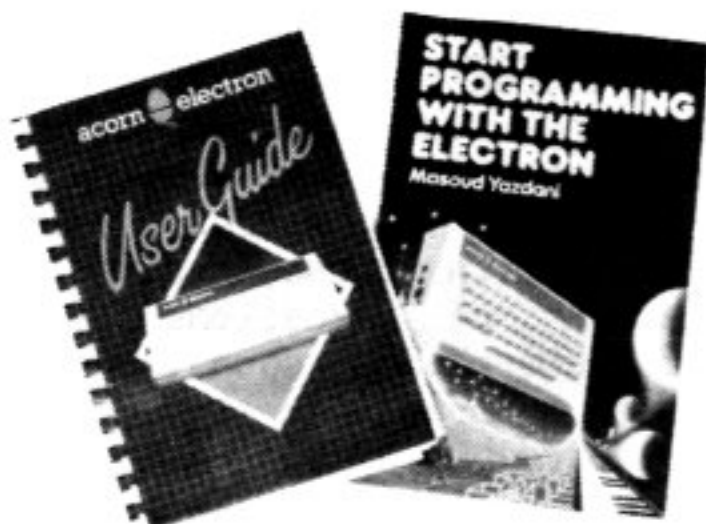


## MACHINE HARDWARE

The machine is well constructed, and has a good feel to it. It has a good quality keyboard of 56 keys, though this is rather noisy in operation. The keyboard allows entry of both upper and lower case characters and includes a number of special keys, such as the



cursor keys, all of which provides flexibility and convenience. A function key (marked FUNC) allows all the number keys to be defined by the user (see article in this issue of ELBUG on Programmable Function Keys) while the same Func key used with the other keys on the keyboard allows direct entry of most Basic keywords. This is a useful facility implemented by the Electron's operating system.



As far as external connections go, the Electron is rather limited as it only carries six connections to the outside world. Somewhat unexpectedly, three of the Electron's sockets are for the visual display (UHF, video and RGB). This leaves the power input socket, a cassette socket (with motor control), and an edge connector to be used with external interfaces. There is no provision for the connection of a printer to the basic machine, possibly the most obvious omission, nor is there any provision to connect disc storage units at present. However, it is unlikely that many Electron users will be bothered by this initially.

Acorn have a number of planned add-ons to upgrade the Electron, but these will not be available for several months, though add-ons from other suppliers have already been announced.

#### INNARDS

Internally, the Electron's circuitry is well laid out. There are two separate circuit boards, one for power supplies and the main computer board containing the CPU or central processor unit (6502A). This board also contains 32k of user memory together with Basic

in ROM, and the Electron's operating system in EPROM. Be warned though, as you will lose at least 11.5k of memory for use by the operating system and for graphics which can grow to as much as 23.5k in some modes. This can cause problems for experienced users but is unlikely to trouble the beginner. See Electron Graphics part 1 for further details.

#### PRINCIPAL FEATURES

The version of Basic used on the Electron is the same BBC Basic used in the BBC micro. This provides, on the Electron, 7 different display modes giving a choice of 20, 40 or 80 characters per line, high, medium or low resolution graphics, and up to 16 colours. These actually consist of six colours plus black and white, all of which can be steady or flashing colours, to give the total of 16. There is also a trade off between the number of colours available at any time and the resolution (the best resolution is 250 points vertically by 640 points horizontally but only allows two colours). Remember too, that the quality of the image will also depend upon the TV or monitor you use with your Electron.

Overall, the Electron really is capable of very good colour graphics, and these facilities are very easy to use from Basic. Even the novice programmer will find it easy to produce interesting and colourful displays. The graphics represent one of the best features of the Electron and so we have included in this issue of ELBUG, the first of a series of articles, telling you how to make the most of your Electron's graphics powers.

A series of commands controls the choice of colour and the drawing of lines and filling of areas. A powerful feature is the ability to define your own graphics characters, which is very useful when writing games programs. There are also two sound channels, controlled by the SOUND and ENVELOPE commands and this is another fascinating feature of the Electron that we shall be talking about in future issues of ELBUG.

BBC Basic also includes all the usual mathematical and string handling



functions plus file processing instructions. One of the most powerful features of BBC Basic is the provision of structured programming support, particularly in the form of procedures. This encourages the writing of clear, readable programs and greatly assists in program development. The book by Yazdani, included in your Electron pack, is a good introduction to this style of programming.

BBC Basic also contains a good 6502 Assembler, and machine code routines are readily incorporated into Basic programs. Your Electron also provides a good screen editor. This is not a full screen editor, as found on some micros, but it is very easy to use indeed. With a program listed on the screen, four cursor control keys are used to move the cursor around on the screen, and a Copy key allows new instructions to be created by copying existing characters from the cursor position. This is a feature which works very well in practice.

#### VALUE FOR MONEY

Although not immediately apparent to the user, one of the economies that has

been made with the Electron is to use an unusual configuration of RAM (Random Access Memory). Because of the way in which this memory is addressed, the Electron's speed of operation varies depending on the amount of memory being used to support graphics. Writing programs for action packed games will need some careful designing to achieve maximum speed. In future issues, we will be introducing you to techniques for getting the best performance out of your Electron.

#### THE COMPETITION

The Electron's competition will be from machines such as the Spectrum, the Dragon and the VIC-20. Broadly speaking the Electron compares extremely favourably with these, offering a combination of excellent text and graphics, two-channel sound, and a full typewriter keyboard. It also has one of the best versions of the Basic language available for a micro and its high compatibility with the very popular BBC micro will add to its attraction at school and in the home. The promised add-ons should prove to be a further incentive in favour of this well designed micro.

## **HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS**

### RND(-TIME)

If you use random numbers in a program and want to make sure that every time you run the program, you get a different random sequence, then put `Z=RND(-TIME)` to initialise the random number generator. TIME is a variable which is automatically incremented every 100th of a second. No matter how fast you can load your program and run it, the sequence of numbers will always be different (to obtain the same sequence a second time, you would need to enter and run the program again from a cold start in exactly the same time, to 100th of a second!).

### FINER POSITIONING OF TEXT

To put text at a specific character position you should use `PRINT TAB(X,Y)`. To position text more finely, to line up with a graphics image for instance, use the command `VDU 5`. Then use `MOVE X,Y` where X and Y are in normal graphics co-ordinates followed by the `PRINT` statement which will display text at the graphics cursor position. `VDU 4` will return to the normal printing procedure.  
E.g. `VDU5:MOVE100,100:PRINT "ELBUG":VDU4`

### SCANNING STRINGS

To scan through all the letters of a string extracting each one in turn use:  
`FOR A=1 TO LEN(A$) : PRINT MID$(A$,A,1) ; : NEXT`  
Here we are simply scanning through the string and printing it out one character at a time. In your own programs you may well want to do something a little more extensive with each character.



## SOUND WIZARD

by Alan Webster

To make the most of the sound capability of your Electron, you need to use two commands, SOUND and ENVELOPE. The sound command is relatively easy to use. It has four parameters Q,A,P,D. Q selects the sound channel number (0 or 1), A switches the sound on or off or selects the envelope, P selects the pitch of the sound, and D the duration. Try the following example:

```
SOUND 1,-15,50,100 <return>
```

This will produce a beep. Increasing the value of the third parameter (50) will increase the pitch, while altering the value of the fourth parameter (100) will alter the duration of the note. You may like to experiment with this a little before progressing. Further details are given in the User Guide p.116.

The ENVELOPE command is more complicated to use, and allows the pitch of a note to be varied while it is being played. With a combination of SOUND and ENVELOPE parameters you can produce a wide variety of different sounds. Putting them together manually is somewhat laborious, and this is where SOUND WIZARD comes in. It provides 8 preset envelopes, and allows you to experiment with these in combination with any chosen set of sound parameters. When you find a combination which sounds good, it allows you to copy the parameters down for use in your own programs. Try typing this program into your Electron and then running it. The computer will ask you to type in values for the sound channel (0-1), the envelope number (1-8), the pitch (0-255) and the duration (0-255). Type a number followed by Return each time.

You should now hear the sound, and the SOUND parameters will be displayed on the screen. You may now enter a new set of parameters, or if you press ESCAPE, the computer will tell you at which program line the appropriate envelope is to be found (for you to transcribe into another program).

Additional envelopes could be added between lines 400 and 500, but you must also include the line number of each envelope in the list at line 210. As only one envelope can be defined at any time, the envelopes have to be redefined when required, using PROCENV(E) where E is the envelope number.

Here are some examples to try:

CHANNEL	ENVELOPE	PITCH	DURATION
1	1	200	40
1	2	200	50
1	3	4	100
0	3	5	40
1	4	50	100
1	5	200	150
1	6	0	10
1	7	20	100
0	8	6	20

```

10 REM Sound Wizard
20 REM Version E1.2
30 REM by A.Webster
40 REM ELBUG November 1983
50 REM Program subject to Copyright
60 ON ERROR GOTO 510
70 E=0
80 MODE 6
90 PRINTTAB(13,0)"SOUND WIZARD"
100 PRINTTAB(12)"by Alan Webster"
110 PRINT'TAB(9)"8 ENVELOPES DEFINED"
120 PRINTTAB(0,5);" Channel Envelope
Pitch Duration"
130 INPUTTAB(1,7),C:INPUTTAB(10,7),E
140 INPUTTAB(20,7),P:INPUTTAB(27,7),D
150 PROCENV(E)
160 SOUND C,1,P,D
170 PRINTTAB(0,16);"SOUND ";C;" ";E;"
";P;" ";D;SPC(100);
180 PRINTTAB(0,7);SPC(40);TAB(0,15);"
ENVELOPE ";E;SPC(5);:VDU30
190 GOTO 90
200 DEFPROCENV(X)
210 ON X GOTO 230,250,270,290,310,330
,350,370
220 GOTO 380
230 ENVELOPE 1,1,-15,-15,-15,230,230,
230,126,0,0,-126,126,126

```



```

240 ENDPROC
250 ENVELOPE 1,1,4,2,-2,5,5,5,126,0,0
,-126,126,126
260 ENDPROC
270 ENVELOPE 1,1,6,0,-6,200,100,200,126
,0,0,-126,126,126
280 ENDPROC
290 ENVELOPE 1,1,-50,-50,-50,20,-20,2
0,126,0,0,-126,126,126
300 ENDPROC
310 ENVELOPE 1,1,-100,100,-100,100,-1
00,100,126,0,0,-126,126,126
320 ENDPROC
330 ENVELOPE 1,1,10,10,10,230,230,230
,126,0,0,-126,126,126
340 ENDPROC
350 ENVELOPE 1,3,0,1,0,0,255,0,126,0,
0,-126,126,126
360 ENDPROC
370 ENVELOPE 1,3,0,1,0,0,255,0,126,0,
0,-126,126,126
380 ENDPROC
390:
400 REM Add Extra Envelopes Here.
500:
510 ON ERROR OFF
520 IF ERR<>17 REPORT:PRINT " at line
";ERL:END
530 IF E=0 PRINT''':END
540 PRINTTAB(0,20);"LIST line ";(20*E
)+210
550 END

```

## HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

### SOUND FREQUENCY - A.J.Vincent

The Sound command is of the form SOUND C,V,F,D where F is the frequency specified as a number in the range 0 to 255. You can use the following function to convert a frequency, measured in Hertz, to the correct frequency parameter for use with the SOUND command:

```
DEF FNfreq(F)=INT(((LOG(F)-LOG(33))*48/LOG(2)-42)+.5)
```

The following short program shows how this function can be used to produce notes of different frequencies:

```

10 INPUT "Please enter the frequency of the sound you
require",frequency
20 X%=FNfreq(frequency)
30 PRINT "The value of the frequency parameter to be used in the SOUND
command is ";X%
40 SOUND 1,-15,X%,30
50 END
60 DEF FNfreq(f)=INT(((LOG(F)-LOG(33))*48/LOG(2)-42)+.5)

```

### GRAPHICS ORIGIN

Don't forget that the graphics origin may be moved by the VDU 29 command:  
VDU 29,640;512;

moves it to the centre of the screen. This is often more convenient for drawing circles and polygons. You can also move the origin more than once in the same program, since the definition of the origin is always taken from the bottom left corner of the screen. VDU 29,0;0; will reset the origin to this default.

This little program repeatedly changes the origin randomly, and draws a square starting at the new origin each time:

```

10 MODE 4
20 VDU 29,RND(1280)-1;RND(1024)-1;
30 MOVE 0,0
40 DRAW 100,0
50 DRAW 100,100
60 DRAW 0,100
70 DRAW 0,0
80 GOTO 20
90 END

```



# ELECTRON GRAPHICS (Part 1)

by David Graham

One of the striking features of the Electron is its graphics capability. We begin here a series of articles exploring Electron graphics.

## WHICH MODE TO USE ?

One excellent feature of the Electron is the wide choice of graphics modes available to the programmer. The disadvantage is that it is not immediately obvious which of the 7 modes to use for any given task. Generally speaking there are five features to watch out for when choosing a mode for a particular task.

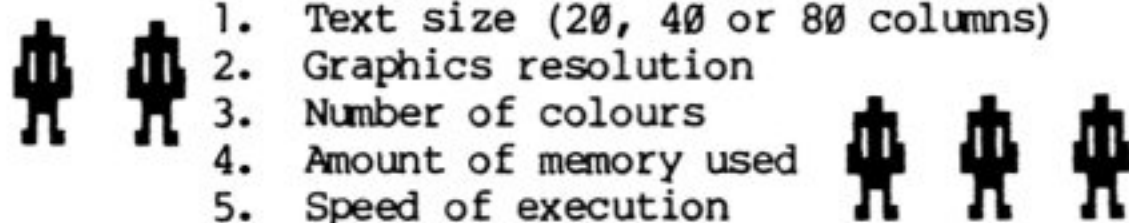
- 
1. Text size (20, 40 or 80 columns)
  2. Graphics resolution
  3. Number of colours
  4. Amount of memory used
  5. Speed of execution

Table 1 gives a breakdown of these features for each mode. As you can see the first 3 factors above are traded off against the fourth and fifth. That is to say, the finer the text required, or the finer the graphics resolution, or the higher the number of colours, the more memory that is used, and the slower the speed of execution.

In fact memory is an extremely

important factor on the Electron, and although the machine is equipped with 32k of RAM (Random Access Memory), the higher resolution modes leave very little for actually storing and running programs. So choosing a mode for anything but the shortest of programs is usually a compromise of the various factors involved.

The most economical overall is mode 6, but this only allows two colours (one background and foreground - eg white on black etc), and only supports text and user defined characters. This is the mode to use if you have a very long program which does not need elaborate screen displays - such as a filing program or word processor. At the other extreme is the very popular mode 2. This uses a great deal of memory, but allows quite high resolution graphics, and most important, it is the only mode to give all 16 colours. It is this mode which is used for most arcade games on the BBC micro, and it is likely that the same will be true of the Electron.

TABLE 1

	Mode	No. of chars.	No. of graphics pixels	No. of colours	Memory used	Remaining memory	Relative speed*
	0	80 x 32	640 x 256	2	20k	5k	11.8s
	1	40 x 32	320 x 256	4	20k	5k	11.8s
	2	20 x 32	160 x 256	16	20k	5k	11.9s
	3	80 x 25	text only	2	16k	9k	9.5s
	4	40 x 32	320 x 256	2	10k	15k	5.6s
	5	20 x 32	160 x 256	4	10k	15k	5.6s
	6	40 x 25	text only	2	8k	17k	5.6s

\*Speed to execute the program TIMETEST. This column can only give a general idea of relative speed, since this varies with the code being executed.

```

10 REM TIMETEST
20 TIME=0
30 FOR A=1 TO 200
40 X=SIN(12)
50 NEXT
60 PRINT TIME

```





Mode 5 can be seen as a compromise between mode 2 and mode 6. Its resolution is the same as that of mode 2, but it only uses half the memory, and gives only 4, rather than 16, colours. The better graphics games in Basic tend to use mode 5 rather than mode 2 simply because good games programs in Basic are usually too long to allow the use of mode 2, whereas machine code programs (most arcade type games are written in machine code) can just squeeze into the available space.

The advantage of mode 4 over mode 5 is the text size. Text in mode 5 (20 characters per line) generally lacks clarity, (try listing a program in this mode!), whereas mode 4 allows full graphics and 40 character text (but only 2 colours).

#### CREATING GRAPHICS.

There are three ways to produce graphics on the Electron:

1. User defined characters
2. PLOT and DRAW line graphics
3. Direct screen access.

The first method uses a facility on the Electron to define characters at will, and then to move them around the screen using PRINT statements or their equivalent. This is well suited to games applications. The second uses a set of graphics commands on the Electron to draw lines and triangles (filled or unfilled), and from these to build up more complex shapes. This is ideal for graphical representations such as pie charts, bar charts and function plotting. In practice many programs use a combination of 1 and 2. The third method can produce similar effects to the first, but it is done by sending codes directly to the screen. It is complicated, and best suited to machine code use, and is generally not recommended for reasons of compatibility with later Electron add-ons.

#### USER DEFINED GRAPHICS

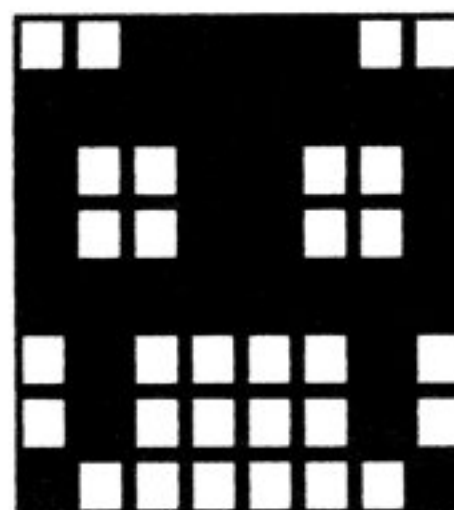
We will begin by looking at the method of User Defined Characters. This is an extremely flexible method, allowing a great variety of displays, and is particularly useful in games writing.

User defined graphics is treated on page 93 of the User Guide, and you should refer to this for further details. It is not treated in Yazdani.

The user defined facility on the Electron allows you to define (or redefine) all of its 256 allowed characters. All characters used on the Electron have an associated number code (0 to 255). The way to print a character from its code is by using the CHR\$ expression. Try

```
PRINT CHR$ 65 <return>.
```

This will print the character 'A' on the screen because the so called ASCII code for A is 65. CHR\$66 will give 'B', and so on. For reasons that will be made clear later, the best character codes to choose for user definition are those from 224 to 255. They are undefined from switch on.



### **Invader**

Try executing PRINT CHR\$224 <return>. Nothing will be printed (at least it won't if you have just switched on). Now type VDU23,224,255,0,255,0,255,0,255,0 <return> then try PRINT CHR\$224 <return> again. It should now give a character with horizontal bars. In fact in the VDU 23 call above (used to define the character 224), each value 255 produced a horizontal line in the block, and each zero, a blank line. The User Guide shows you more fully how each of the last 8 parameters of the VDU 23 call are made up to produce any given shape. The first parameter of all (224 in this case) specifies which character number is to be redefined.

To take the mental strain out of all this, our ELBUG introductory cassette contains a program called Redefine. If



VDU23, #, 60, 255, 153, 153, 255, 66, 66, 129.

You may like to try to produce your own characters. As a further example page 114 of the User Guide gives the 4 card suit shapes. You might also like to define an alternative character set, either for foreign alphabets - eg the accented letters of French, or a different style alphabet from the one provided.



To place a character at any chosen position on the screen, use the TAB statement. For example, the statement:

will place your character 10 spaces from the left on the 20th line down; always assuming that character 224 has been previously defined of course.

```

100 REM MOVING INVADER
110 MODE4
120 VDU23,224,60,255,153,153,255,66,6
6,129
130 FOR A%=1 TO 38
140 PRINT TAB(A%,10) CHR$224
150 NEXT
160 END

```



```
145 TIME=0: REPEAT UNTIL TIME=10
```

```
140 PRINT TAB(A%,10)SPC1;CHR$224
```

Next month we will look more closely at user defined graphics, and will introduce techniques for moving larger characters, for producing multicoloured characters, and for animating them.

## BREAK AND CONTROL BREAK

Pressing Ctrl-Break (i.e. pressing Control and Break together) has a similar effect to pressing Break alone, except that more of the internal settings are returned to their starting values. The most important of these are the key definitions which are cleared completely. Thus programs which define the Break key (function key 10) to restart themselves may be stopped by pressing Ctrl-Break. The OLD command will still restore the current program in memory.

Typing \*FX210,1 <return> will switch off all sound output. Do this before loading a program from cassette. To re-enable sounds either press Break or type \*FX210,0 <return> after the program has finished running.



Enlarge your text displays with this useful utility. It will really make your programs stand out.

Large size text on the screen can add impact to your programs, but with the Electron there is no simple facility for generating double height characters. To overcome this we present a procedure which allows double height printing in any of the graphics modes. This is extremely useful for making more prominent displays, often needed in education and for games. The procedure, called PROCdouble, is defined in lines 1000 to 1120 and is very easy to use. Simply add it on to the end of your program, and it can then be called as a normal procedure. For example

```
PROCdouble("FORTY-TWO",5,10)
```

will display "FORTY-TWO" in double



```
10 REM Program DOUBLE
20 REM Version El.1
30 REM Author Bjorn Grand
40 REM ELBUG November 1983
50 REM Subject to COPYRIGHT
100 MODE 2
110 COLOUR 6:COLOUR 128+4
120 PROCdouble("ELBUG",6,3)
130 PROCdouble("for the",6,9)
140 PROCdouble("ELECTRON",6,15)
150 END
1000 DEF PROCdouble(A$,K,L)
1010 LOCAL N
1020 A%=&A:X%=0:Y%=&A
```

BJORN GRAND'S

# DOUBLE HEIGHT CHARACTERS

height characters, and in a position on the screen, 10 characters down from the top and 5 characters in from the lefthand side. The string can be a string variable, and the co-ordinates can also be variables.

We have put the procedure PROCdouble following a short program in lines 10 to 150 to show you how to use it. This program calls the procedure three times in order to display three lines of double height text on the screen. Why not type the program into your

Electron and then run it to see the results?

If you want to use this procedure in one of your own programs, you are recommended to consult the Electron User Guide, pages 200 and 201, which tell you how to merge Basic programs together.

```
1030 D=&A00
1040 FORN=1 TO LEN(A$)
1050 B$=MID$(A$,N,1)
1060 ?D=ASC(B$)
1070 CALL (&FFF1)
1080 VDU23,240,D?1,D?1,D?2,D?2,D?3,D?3
,D?4,D?4
1090 VDU23,241,D?5,D?5,D?6,D?6,D?7,D?7
,D?8,D?8
1100 PRINT TAB(K+N,L);CHR$(240);TAB(K+
N,L+1);CHR$(241)
1110 NEXT N
1120 ENDPROC
```

This program was adapted from a version for the BBC micro first published in BEEBUG Vol.2 No.1.

The same procedure is used in the MENU program on the introductory ELBUG cassette. It is situated there, in lines 500 to 730. You may find it quicker to load the MENU program from cassette, delete lines 10 to 490 and simply type in from the magazine listing, lines 10 to 150.



## HIGHLO CARD GAME

by R Bailey

HIGHLO is a simple card game, but one which rapidly becomes quite addictive. It is based on the TV show, 'Play Your Cards Right'. You are presented with 9 cards face down on the screen. The first card is turned over and you have to bet on whether the next card, when turned over, will be higher or lower in value. If you reach or exceed the target of 2500 points with each set of cards, then you continue with a new set of cards. With the first, fourth and seventh cards, you also have the opportunity to change the card before placing your bet.

With each card you have to decide how much to bet, and you will need to think carefully how best to do this, depending on the face value of the card displayed, if you are to reach or exceed the target.

The screen presentation is excellent and contains full user prompts throughout for playing the game. If you enjoy games of chance then this is the game for you.

```

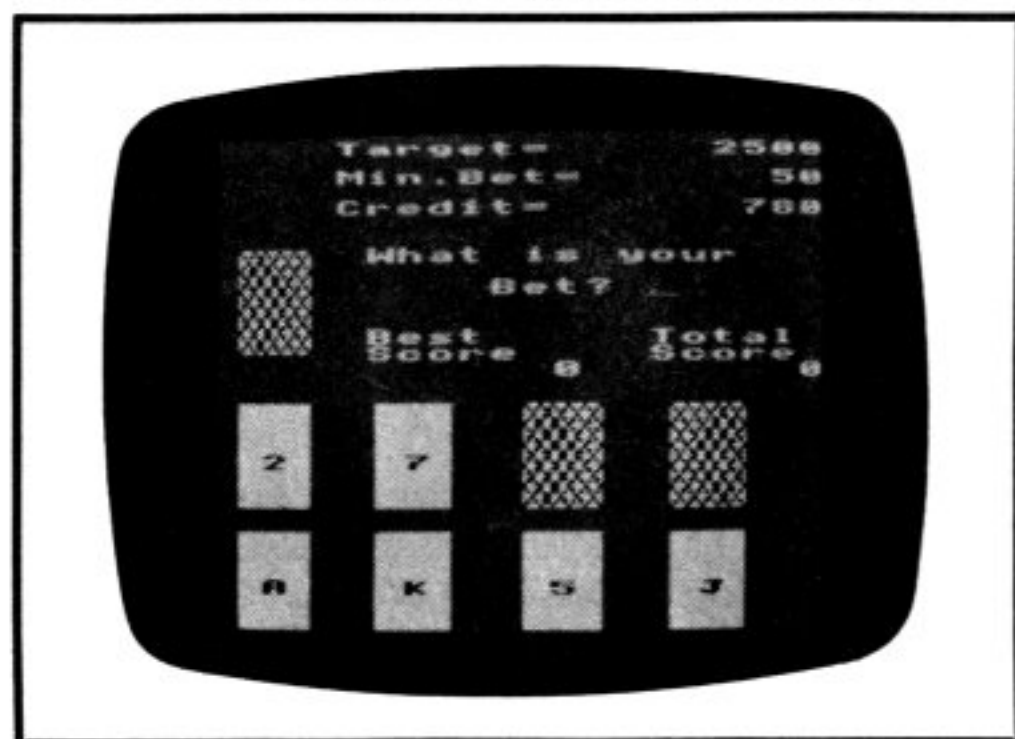
10REM Program HIGHLO
20REM Version El.2
30REM Author R.Bailey
40REM ELBUG November 1983
50REM Program subject to Copyright
60ON ERROR GOTO 1060
70X=RND(-TIME)
80VDU23,224,206,219,219,219,219,219,
206,0
90MODE5:DIMA(12),B$(9),S(12):@%=&000
00908
100BS=0
110VDU28,4,16,19,0:VDU19,128,4,0,0,0,
19,2,0,0,0,0
120RT=0:CLS
130C=250:MB=50:AIM=2500
140F=0:PROCT:RESTORE:FORJ=1TO9:READX,
,Y:PROCC:NEXT
150 PROCs:PROCV:RESTORE:F=1
160 FORJ=1TO9:READX,Y:PROCC:PROCSh:IF
J=9THEN200
170 IFC<MB J=9:GOTO210
180 IFJ=1ORJ=4ORJ=7THENPROCCH
190 PROCQ:PROCU:PROCHL
200 NEXT
210 CLS:RT=RT+C:IFC>=AIM THEN330
220 IFRT>BS THENBS=RT
230 IFC<MB THEN310
240 PRINTTAB(0,2)"You have failed"
250 PRINTTAB(1,4)"to reach the"
260 PRINTTAB(4,6)"TARGET"
270 PROCBS
280 PRINTTAB(0,8)"ANOTHER GAME?"
290 PRINTTAB(4,10)"Y/N":A$=GET$
300 IFA$="Y"THEN120ELSE CLG:PRINTTAB(
0,10)"SEE YOU SOON":TIME=0:REPEAT:UNTIL
TIME=200:MODE6:END
310 PRINTTAB(0,2)"You have failed"
320 PRINTTAB(2,4)"miserably":GOTO270
330 PRINTTAB(0,9)"CONGRATULATIONS":GO
TO130
340 DATA50,10,350,10,650,10,950,10
350 DATA50,260,350,260,650,260,950,26
0,50,560
360 DEFFNX=150-Z*30
370 DEFFNY=200-Z*40
380 DEFPROCC
390 GCOL0,3
400 MOVEX,Y:DRAWX,Y+200:DRAWX+150,Y+2
00:PLOT85,X,Y
410 DRAWX+150,Y:DRAWX,Y:PLOT85,X+150,
Y+200:IFF=1THENENDPROC
420 GCOL0,1:FORZ=0TO4
430 MOVEX,Y+Z*40:PLOT1,FNX,FNY
440 MOVEX+Z*30,Y:PLOT1,FNX,FNY
450 MOVEX,Y+FNY:PLOT1,FNX,-FNY
460 MOVEX+Z*30,Y+200:PLOT1,FNX,-FNY
470 NEXT:ENDPROC
480 DEFPROCS
490 FORJ=1TO12

```

```

500 A(J)=RND(52):F=0
510 IFJ=1THEN540
520 FORK=1TOJ-1:IFA(J)=A(K)THENF=1
530 NEXT
540 IFF=1THEN500
550 NEXT
560 FORJ=1TO12:FORL=1TO4
570 IFA(J)>13THENA(J)=A(J)-13:S(J)=L
580 NEXT:NEXT:ENDPROC
590 DEFPROCCH
600 PROCD:PRINTTAB(2,8)"Change card?"
610 PRINTTAB(6,10)"Y/N":REPEAT:A$=GET$

```



```

620 UNTILA$="Y"ORA$="N"
630 IFA$="N"THEN680
640 IFJ=1THENA(J)=A(10):S(J)=S(10)
650 IFJ=4THENA(J)=A(11):S(J)=S(11)
660 IFJ=7THENA(J)=A(12):S(J)=S(12)
670 PROCV:PROCC:PROCSH
680 PROCD:ENDPROC
690 DEFPROCV
700 FORL=1TO9
710 B$(L)=CHR$(A(L)+49)
720 IFA(L)=9THENB$(L)=CHR$224
730 IFA(L)=10THENB$(L)="J"
740 IFA(L)=11THENB$(L)="Q"
750 IFA(L)=12THENB$(L)="K"

```

```

760 IFA(L)=13THENB$(L)="A"
770 NEXT:ENDPROC
780 DEFPROCSH
790 VDU5:IFS(J)=1ORS(J)=3THENGCOL0,1E
LSEGCOL0,2
800 MOVEX+50,Y+100
810 PRINTB$(J):VDU4:ENDPROC
820 DEFPROCT
830 PRINTTAB(0,1)"Target=",AIM
840 PRINTTAB(0,3)"Min. Bet=",MB
850 PRINTTAB(0,5)"Credit=",C
860 PROCBS:ENDPROC
870 DEFPROCQ
880 REPEAT:PRINTTAB(0,8)"What is your
bet"
890 INPUTTAB(2,10)"Bet= "B:PROCD:UNTI
LB>=MB ANDB<=C
900 C=C-B:PROCD:ENDPROC
910 DEFPROCHL
920 PRINTTAB(0,8)"Higher or lower"
930 REPEAT:PRINTTAB(5,10)"H/L...":A$=
GET$:UNTILA$="H"ORA$="L"
940 IF(A$="L"ANDA(J+1)<A(J))OR(A$="H"
ANDA(J+1)>A(J))THENC=C+B*2:SOUND1,-10,2
00,10 ELSE SOUND 1,-10,20,10:SOUND1,-10
,1,10
950 PROCD:PROCU:ENDPROC
960 DEFPROCD
970 PRINTTAB(0,7)SPC(80):ENDPROC
980 DEFPROCU
990 PRINTTAB(8,5)SPC(8):PRINTTAB(8,5)
,C
1000 ENDPROC
1010 DEFPROCBS
1020 PRINTTAB(1,13)"Best      Total"
1030 PRINTTAB(1,14)"Score    Score"
1040 PRINTTAB(0,15),BS,RT;
1050 ENDPROC
1060 ON ERROR OFF
1070 IF ERR=17 A$="":GOTO 300
1080 MODE6
1090 REPORT:PRINT" at line ";ERL
1100 END

```

.....  
 This program was adapted from a version for the BBC micro published in BEEBUG Vol.1 No.5.

## HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

### LOGICAL/PHYSICAL COLOURS

When using commands such as COLOUR and GCOL you are setting the LOGICAL colour in which these actions occur on the screen, and this is the colour that POINT will return.

The command VDU19,logical,physical,0,0,0 sets up which physical colour the video generation circuitry sends to the television screen when the video system reads through the memory to make up the picture. The 16 physical colour numbers are as listed on page 108 of the Electron User Guide.



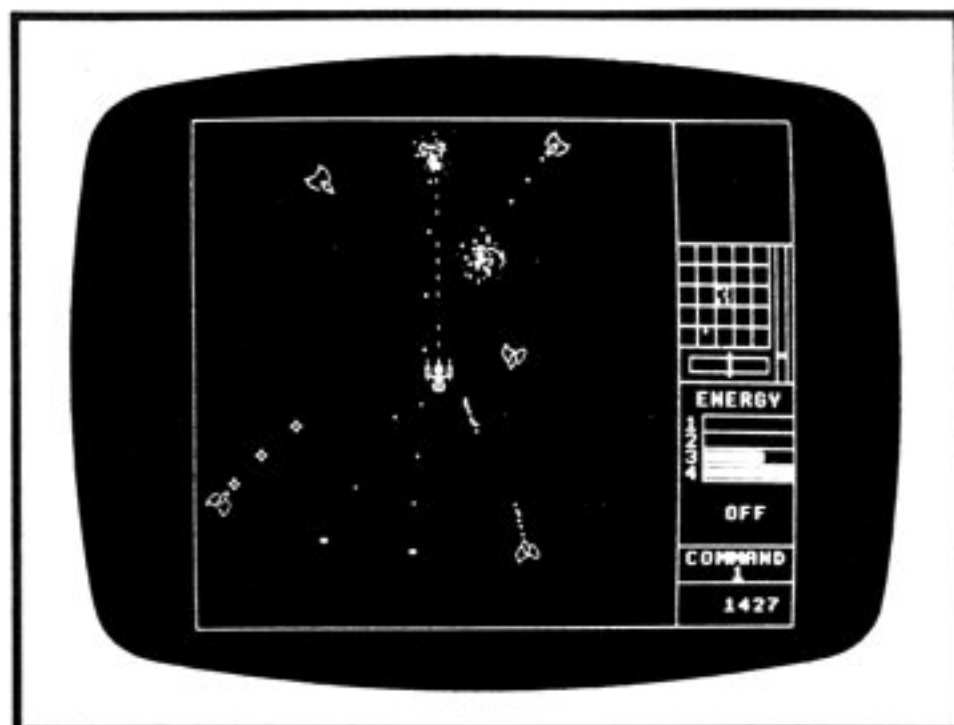


# THE FIRST ELECTRON SOFTWARE FROM ACORNSOFT

Reviewed by Philip Le Grand

We review here the first of the new software from Acornsoft which has been written for the Electron or adapted from previous BBC micro versions, ready for the Electron's release.

Let us start with some general comments, since the software is all from the same publisher, Acornsoft. This software is generally of a high standard and has been well and attractively packaged for the Electron. Each program comes with documentation, usually as a separate leaflet or booklet, which is also of good quality. There are some references here, to using printers and joysticks, although the basic Electron cannot handle these at the moment. It is also worth noticing that all of the software reviewed here has previously been implemented on the BBC micro, and that one or two of the points raised later may be the result of their adaption to the Electron.



This is an entirely new arcade style game, first developed for the BBC micro. It makes good use of graphics though only in two colours. You are in a starship and you have a visual read-out of position of enemy ships and your energy levels on the right of the screen. Your ship stays in the centre of the screen in the same direction so when you bank left or right, the rest or the universe rotates around you. There are many controls to get used to, all giving extra interest. This is a good game which should sustain your interest for quite some time.

Name : METEORS  
Price : £9.20 inc. VAT  
Rating : \*\*\*

In this game, you control a spaceship in the centre of the screen, which may be rotated left or right, and moved forwards with the thrust control. Your task is to destroy meteorites which float around generally in your direction. If you are hit by them, your ship is destroyed (silently!). If you shoot a meteorite, it divides into smaller parts which still follow you and become harder to destroy. Occasionally a UFO enters the field of view and you score more points by destroying it. If you get into a tight situation you can enter hyperspace and re-emerge anywhere.

In principle, this is a good game, but this implementation is not very exciting. The action is rather slow and the graphics are purely black and white. For the price of £9.20, the other games reviewed here offer better value for money.

Name : Starship Command  
Price : £9.20 inc. VAT  
Rating : \*\*\*\*

Name : MONSTERS  
Price : £9.20 inc. VAT  
Rating : \*\*\*\*

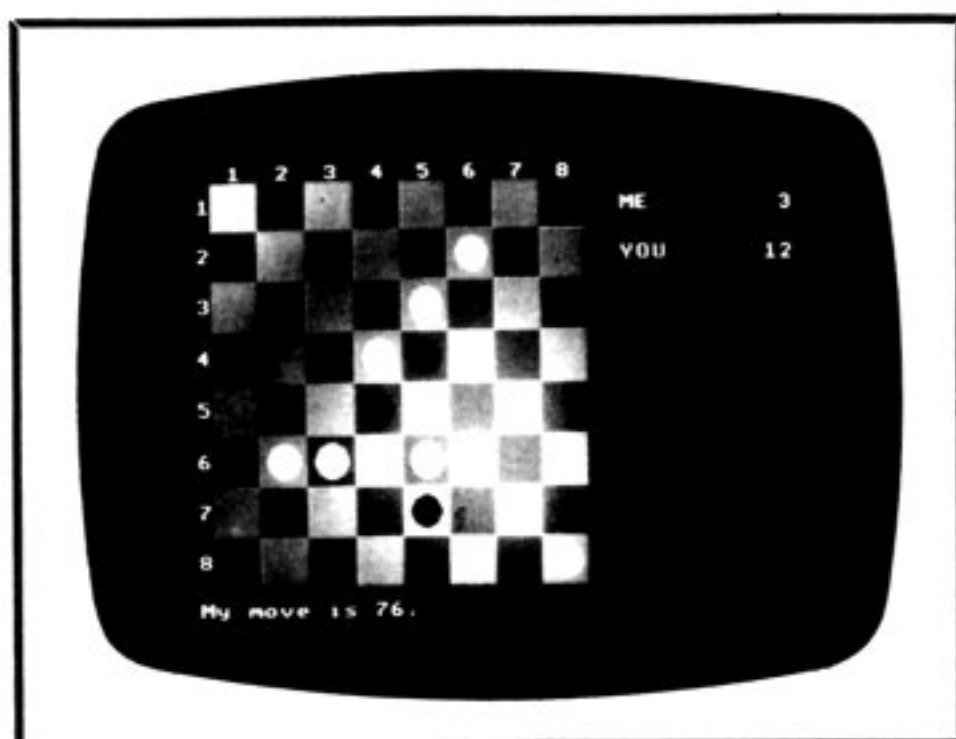
This is an excellent game for showing off the quality of the Electron's colour graphics and is accompanied by a nice variety of computer generated sound effects. You control a man who can run across floors and up and down ladders to other floors, in order to avoid being eaten by the multicoloured monsters. When you find a quiet spot, you dig a hole and hope that a monster falls in. When it does, you can destroy it by filling in the hole again before it can escape.



There are different types of monsters, each requiring slightly different tactics.

The more monsters that appear, the slower the game, since the screen handling is all in software on the Electron. This is rather disconcerting as in the course of playing, the game appears to slow down with each new frame and then speed up as the monsters are killed. This is more a game of tactics and strategy rather than fast manoeuvres and quick reaction. It is quite a good program and has been popular on other micros.

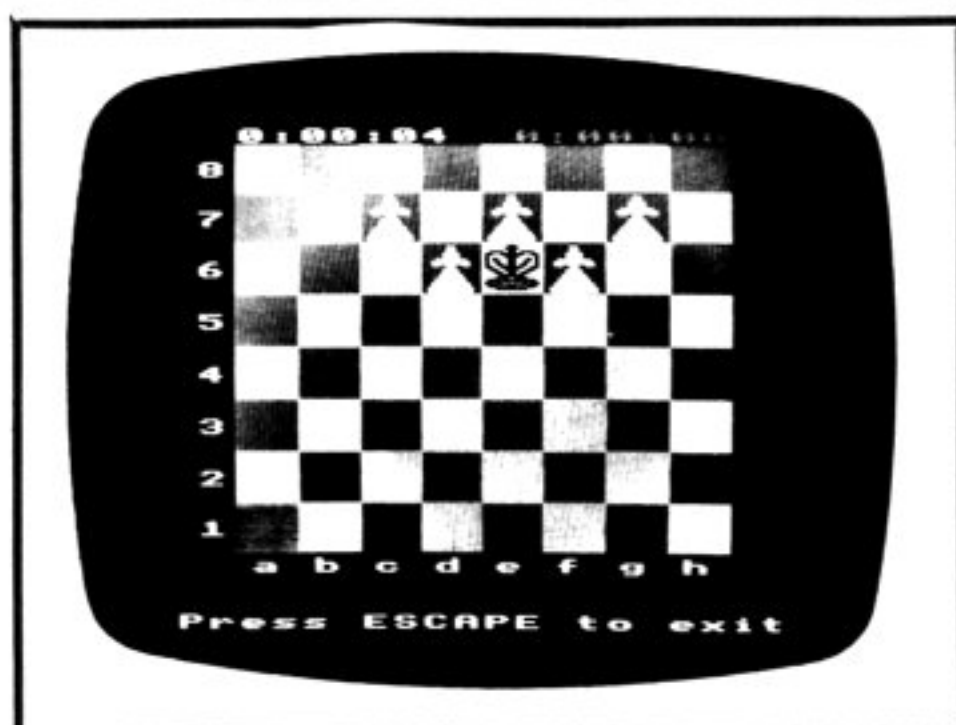
~~~~~  
 Name : DRAUGHTS & REVERSI  
 Price : £9.20 inc. VAT  
 Rating : \*\*\*\*



There is good value for money here, since you get two games in this package for the price of one. The board and pieces are well displayed, giving a 3D effect to the pieces. Both games offer a series of different playing levels but although the computer takes but a few seconds to make its move at the lowest level, at the other extreme you may have to wait up to 45 minutes.

Moves are entered in a co-ordinate form. This pack and the chess pack are good packs to buy if you are fed up with the usual 'seek-em-and-splatt-em' type of games and want something to get your brain into.

~~~~~  
 Name : CHESS  
 Price : £9.20 inc. VAT  
 Rating : \*\*\*\*



The representation of each of the pieces is very good, allowing clear identification of each piece on the board. The program allows several options, some of which are:-

Auto play, where the computer plays itself, the choice of playing black or white against the computer or the computer acting like a real board so that you can play against another opponent. In each mode there is an on-screen clock ticking away and timing the moves, though you can turn the sound off if you want to.

Moves are entered using the usual chess notation. It has 10 levels of skill, taking between 4 seconds and over 7 hours for the first move! (We couldn't bear to continue playing like this!). Quite obviously the program does not use set openings.

~~~~~  
 Name : PERSONAL MONEY MANAGEMENT  
 Price : £11.50 inc. VAT  
 Rating : \*\*\*\*

This is a useful money management package. The program initially assumes that you are an average person with a Current and Deposit account with the following expenses:-

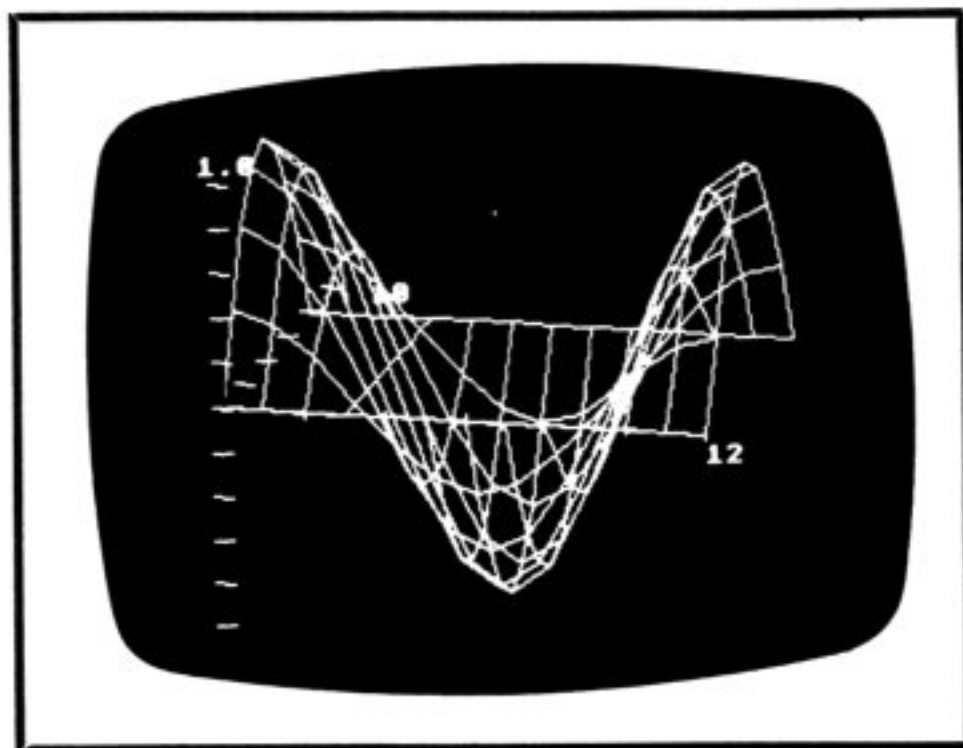
Car/travelling costs, electricity, gas/oil, insurance, mortgages, rates, subscriptions, telephone, standing orders and miscellaneous expenses. You can then add or delete articles as you wish, to tailor the package to your own requirements. This program handles the data for a whole year in monthly units and can give information displays of current savings, budgets, next month's





spending, annual budgets and spending over the last 12 months. All entered data may be saved onto tape and re-entered at any time.

Name : GRAPHS & CHARTS  
& CREATIVE GRAPHICS  
Price : both cost £9.20 inc. VAT  
Rating : \*\*\*\*



These two packs consist of a set of very useful programs and procedures. Both are adapted from earlier versions for the BBC micro for which listings of the programs, plus useful explanations of the techniques involved, were also published as books with the same titles for £7.50 each.

The Graphs and Charts package allows you to produce most types of graph or chart you are ever likely to

use, e.g. bar charts, x,y graphs, curve or function plotting, pie charts, 3D curves, 2D contour maps, 3D contour maps and surfaces. The programs are stand-alone units and are useful in, say, educational applications, for producing velocity-time graphs, distribution graphs, colony count charts, etc, or for teaching the use of different methods for displaying data. The procedures can all be included in your own programs for more specific applications.

The graphics covered in the Creative Graphics pack, are mainly plotting graphics, although user defined characters are briefly touched on. Included are methods for drawing a variety of mathematical figures, techniques for movement and animation, and some interesting routines for representing 3D objects. There are also some interesting mathematical curves and a number of other patterns, such as the CARPET program, all of which which produce some very striking effects. The final programs show various scenes which incorporate techniques developed earlier in the book. Both packs can be recommended, particularly for those interested in developing their own programs of this kind.

Supplier: All Acornsoft software can be ordered from Vector Marketing, Dennington Industrial Estate, Wellingborough, Northants, NN8 2RL. Many computer shops and dealers also stock Acornsoft software.

## HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

### CONTROL KEY CODES

Some of the more useful control codes, generated by holding down the CTRL key at the same time as pressing certain keyboard letters, are as follows:

|        |                                                     |
|--------|-----------------------------------------------------|
| Ctrl-G | Beep (same as VDU7)                                 |
| Ctrl-H | Cursor left (non-erasing)                           |
| Ctrl-I | Cursor right                                        |
| Ctrl-J | Cursor down                                         |
| Ctrl-K | Cursor up (scrolls screen once top line is reached) |
| Ctrl-L | Clear text screen                                   |
| Ctrl-M | Return                                              |
| Ctrl-N | Turns on paging mode                                |
| Ctrl-O | Turns off paging mode                               |
| Ctrl-P | Clears graphics screen                              |

### HEX TO DECIMAL AND BACK

PRINT ~14 prints the hexadecimal equivalent of 14 i.e. E

PRINT &20 prints the decimal equivalent of 20 (hex) i.e. 32



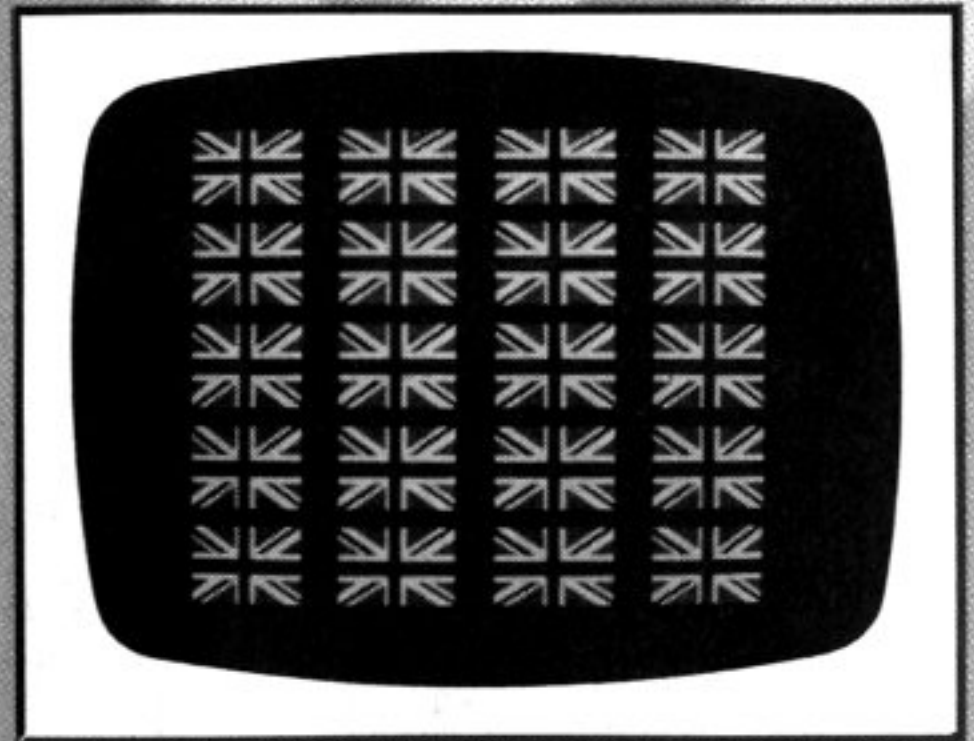
# UNION JACK

by David Stonebanks

This program produces a whole series of representations of the Union Jack with various aspect ratios. The display is colourful and eye-catching, and gives a good demonstration of the quality of the Electron's graphics.

Pressing the spacebar down during the running of the program, will freeze the display at the end of the frame in progress. Pressing the spacebar again will release it.

This program was adapted from a version for the BBC micro first published in BEEBUG Vol.1 No.6.



```

10 REM Program JACK
20 REM Version E1.1
30 REM Author David Stonebanks
40 REM ELBUG November 1983
50 REM Subject to Copyright
100 ON ERROR ON ERROR OFF:MODE6:REPOR
T:PRINT" at line ";ERL:END
110 REPEAT
120 MODE 1
130 VDU23,1,0;0;0;0;0:REM delete cursor
140 black=0:red=1:blue=2:white=3
150 VDU19,blue,4;0;
160 VDU29,640;512;:REM set graphics
170 FOR hsize=5 TO 60 STEP 10
180 vsize=hsize:REM square
190 PROCunionjack
200 PROCdelay(100)
210 NEXT hsize
220 CLS
230 FOR hsize=5 TO 60 STEP 10
240 FOR vsize=5 TO 60 STEP 10
250 PROCunionjack
260 PROCdelay(100)
270 NEXT vsize
280 CLS
290 NEXT hsize
300 CLS
310 hsize=10:vsize=10
320 FOR X=160TO1200 STEP 320
330 FOR Y=100 TO 900 STEP 200
340 VDU29,X;Y;:REM move graphics origin
350 PROCunionjack
360 NEXT Y
370 NEXT X
380 PROCdelay(200)
390 UNTIL FALSE
400 END
1000 DEFPROCunionjack
1010 PROCbox(blue,12,8)
1020 PROCdiag(white,9,8,-12,-6,12,8,-1
2,-8)
1030 PROCdiag(white,12,8,-12,-8,12,6,-
9,-8)
1040 PROCdiag(white,-9,8,-12,8,12,-6,1
2,-8)
1050 PROCdiag(white,-12,8,-12,6,12,-8,
9,-8)
1060 PROCdiag(red,10.5,8,0,1,12,8,0,0)
1070 PROCdiag(red,0,0,-12,-8,0,-1,-10.
5,-8)
1080 PROCdiag(red,0,0,-12,8,0,-1,-12,7
)
1090 PROCdiag(red,0,0,12,-8,0,1,12,-7)
1100 PROCbox(white,2,8)
1110 PROCbox(white,12,2.5)
1120 PROCbox(red,1,8)
1130 PROCbox(red,12,1.5)
1140 IF INKEY-99 THEN REPEAT UNTIL NOT
INKEY-99:REPEAT UNTIL INKEY-99
1150 ENDPROC
1160 DEFPROCbox(colour,halfx,halfy)
1170 GCOL0,colour
1180 MOVEhalfx*hsize,halfy*vsize:MOVE-
halfx*hsize,halfy*vsize
1190 PLOT85,halfx*hsize,-halfy*vsize:P
LOT85,-halfx*hsize,-halfy*vsize
1200 ENDPROC
1210 DEFPROCdiag(colour,x1,y1,x2,y2,x3
,y3,x4,y4)
1220 GCOL0,colour
1230 MOVEx1*hsize,y1*vsize:MOVEx2*hsiz
e,y2*vsize
1240 PLOT85,x3*hsize,y3*vsize:PLOT85,x
4*hsize,y4*vsize
1250 ENDPROC
1260 DEFPROCdelay(t)
1270 T=TIME+t:REPEAT UNTIL TIME>T
1280 ENDPROC

```



## USING THE PROGRAMMABLE FUNCTION KEYS

by Mike Williams

The special offer cassette sent to new ORBIT members contains a program called KEYSET for programming the function keys on the Electron. Now we explain all about this useful feature and the KEYSET program.

If you have read about the use of the Func key in the Electron User Guide (see pages 19, 42 and 43) you will know that you can use this with the other keys on the keyboard to generate directly, many of the Basic keywords. For example, pressing Func/R (i.e. the key marked 'FUNC' and the key marked 'R' together) will produce 'RUN' and is equivalent to typing the same word letter by letter but, of course, much quicker. You will also notice, that the fronts of the numeric keys (1 to 0) are marked f1, f2 etc. However, if you press for example, Func/f1, nothing happens. This is because these keys do not have any built in functions, but you can program them to produce any character or character string that you want.

How useful this can be is indicated by the KEYSET program on the introductory ORBIT cassette. For example, Func/f8 prints out the amount of free memory left in your machine at any time, and so on. The memory display utility, MEMDISP, also on the same cassette, is another program that uses function keys.

To program a function key, say function key f0, you type

```
*KEY0
```

and follow this, on the same line, with the string of characters that will define the function of that key. For example, suppose you would like to have key f0 set mode 6. Try typing in the following, and then press Func/f0

```
*KEY0 "MODE 6" <return>
```

You should find that the words 'MODE 6' appear on the screen but that nothing else happens. This is because using the function key is just like typing directly on the keyboard, though much

quicker because only two keys have to be pressed. When we defined key f0 we did not include a Return (the key marked 'RETURN') at the end of 'MODE 6' as we would when typing it directly from the keyboard. You can complete the command by typing the Return from the keyboard, but it would be better if we could include it in the definition of the function key itself. There is a special way of including Return and other control codes. This uses the '|' character followed by one other character. The '|' character is on the key next to Break. You can find a list of all the control codes in Appendix A of the Electron User Guide. If you look at this, you will see that CTRL/M is exactly the equivalent of Return and this is what we need. So our revised definition for key f0 is now

```
*KEY0 "MODE 6|M" <return>
```

Type this in and then press Func/f0. Because we have now included the Return, the computer obeys this command straight away, and the screen flashes as it always does when you change mode.

We can set up a function key to do more than just one simple task. Suppose we would like key f0 to not just set mode 6, but also set paged mode (VDU14 or CTRL/N) and issue the command LIST. We can do this as follows

```
*KEY0 "MODE 6|M|N LIST|M" <return>
```

Try typing this on the keyboard of your Electron and then press Func/f0 to see the effect. Of course, unless you have a Basic program in memory at the time, there will be nothing to list on the screen.

In the example above, key f0 has been set up to produce the characters enclosed by quotes. The quotes are not essential, but help to make clear the

beginning and end of the string. If you omit the character string itself, then that clears any existing definition for that key. Function key definitions can also be included as lines in a program.

#### THE KEYSET PROGRAM

One good way of using the function keys, is to set them up to do all the things you are likely to need when loading, saving and developing programs. A program to do this, called KEYSET, was supplied on the introductory cassette that you received when you joined ORBIT. If you load and run this program (CHAIN"KEYSET" or select from cassette menu), it defines all the function keys and displays a list on the screen to tell you what each function key will do. Of course, this information will not be preserved once you start using the function keys. To help you remember the key functions, we have included in the magazine, a strip for you to cut out or copy, which you can place on the top of your Electron.

The program listing, shown below, will also help you to understand further, some of the techniques to be used in defining function keys.

There are two additional points to note in the listing. Many Basic keywords can be abbreviated (e.g. P. instead of PRINT in line 260) to save space. This information is not given in the Electron User Guide so we shall be publishing this in a future issue of ORBIT. Secondly, a function key definition can contain several Basic instructions separated by the ':' character as shown in line 280. This can also save space.

You will also notice a few minor differences between the version of KEYSET on cassette and that listed here in ELBUG. These are not significant but help to improve the readability of the magazine version. This is the version referred to in this article, since the line numbering differs.

```

10 REM Program KEYSET
20 REM Version E1.2
30 REM ELBUG November 1983
40 REM Program subject to Copyright
50 MODE6
60 VDU19,0,1,0,0,0
70 *FX18
80 PRINTTAB(10)"ORBIT KEY SET ROUTINE"
90 PRINT""F1 - Load a program from cassette"
100 PRINT"F2 - Chain a program from cassette"
110 PRINT"F3 - Verify a program"
120 PRINT"F4 - Display a catalogue of programs "" on a cassette (and turn motor on)"
130 PRINT"F5 - Turn cassette motor off"
140 PRINT"F6 - Mode 6 colour display"
150 PRINT"F7 - The length of a program (in decimal)"
160 PRINT"F8 - The amount of free memory (in decimal)"
170 PRINT"F9 - Line listing after an error"
180 PRINT"F0 - Formatted listing-Shift to continue"
190 PRINT"For further details :-"" see the first issue of ELBUG."
200 *KEY1 LOAD""|M
210 *KEY2 CHAIN""|M
220 *KEY3 *LOAD"" 8000|M

```

|                    |        |
|--------------------|--------|
| ORBIT<br>KEYSET    |        |
| FORMAT<br>LIST     |        |
| ERROR<br>LIST      |        |
| FREE<br>MEMORY     |        |
| PROGRAM<br>LENGTH  |        |
| MODE 6<br>DISPLAY  |        |
| MOTOR<br>OFF       |        |
| CAT<br>MOTOR<br>ON |        |
| CASSETTE           | VERIFY |
|                    | CHAIN  |
|                    | LOAD   |
|                    | KEYSET |
| ORBIT              |        |



```

230 *KEY4 *CAT|M
240 *KEY5 *MOTOR0|M
250 *KEY6 MO.6|M VDU19,0,4,0,0,0|M CL
S|M
260 *KEY7 P.TOP-PAGE|M
270 *KEY8 DIM P%-1|M P.HIMEM-P%|M

```

```

280 *KEY9 C$="LIST"+STR$ERL+CHR$13:A%
=138:X%=0:F.L=1TOLENC$:Y%=ASC(MID$(C$,L
)):CA.&FFF4:N.|M
290 *KEY0 MO.6|M LIST07|M L.|N|M LIST
00|M
300 *FX4,0
310 END

```

# MORE ON FUNCTION KEYS

We conclude this article with some more information about defining and using function keys.

The function key definitions that you specify are stored in a reserved area of memory, 256 bytes long and starting at memory address &B00. If you have programmed some, or all, of the function keys, you can save the definitions on tape by typing:-

```
*SAVE "KEYS1" B00+100 <return>
```

where KEYS1, or whatever name you choose, is the name of the file on tape. You can then load these definitions back into the Electron at any time, without disturbing any Basic program that you happen to have in the machine, by simply typing

```
*LOAD "KEYS1" <return>
```

Of course, this only programs the function keys. No information as to their use will be displayed on the screen as happens when you run the KEYSET program.

Because there is only a limited amount of space in which to store function key definitions, it is often necessary to clear out existing definitions before redefining. This can be done individually as previously described, but \*FX18 will clear all existing function key definitions.

The Break key can also be programmed as function key 10, in just the same way as the other function keys. If you then press Break, the machine will carry out a reset as usual, followed by the user programmed definition. A very useful thing to do here, is to program the Break key as follows:-

```
*KEY10 "OLD|M" <return>
```

Thus if you accidentally press the Break key the machine will immediately follow reset with the OLD command, restoring your program before any further damage occurs. You will find another example of programming the Break key on page 131 of the Electron User Guide. Similarly, the four cursor control keys and the Copy key can also be re-programmed as keys 11 to 15, though this could be confusing and is thus not to be recommended unless essential.

**IF you are not already a member of ORBIT,  
Subscribe now, and get a free introductory cassette  
containing 8 tested programs for the Electron.**

1. **SPACE CITY.** Defeat the invading Aliens with your laser, and save the city
2. **3D NOUGHTS AND CROSSES.** Pit your wits against the ELECTRON on a 4x4x4 board
3. **RACER.** Guide your racing car to victory, avoiding other cars and obstacles on the track
4. **3D MAZE.** In this challenging game, you must escape from the maze — The screen displays a 3D view from inside the maze
5. **PATCHWORK.** A multicoloured display of continuously changing patterns
6. **KEY SET ROUTINE.** A program to set up the user function keys
7. **MEMORY DISPLAY.** An efficiently written utility to display the contents of memory (ROM and RAM)
8. **CHARACTER DEFINER.** Define individual graphics characters with this useful utility for use in your own programs

See back cover of the magazine for subscription details



SPACE  
CITY



RACER



BEEBMAZE

## THREE TAPE RECORDERS FOR THE ELECTRON

Reviewed by Matthew Rapier

The Electron is a brand new machine. One of the first peripherals that you will need, to make full use of it, is a good cassette recorder. In this review, Matthew Rapier tests three readily available recorders on the Electron, and reports on his findings.

### MACHINES REVIEWED

|               |               |              |
|---------------|---------------|--------------|
| 1. BOOTS      | CR325         | PRICE £23.95 |
| 2. W.H. SMITH | CCR800        | PRICE £29.95 |
| 3. ACORN      | DATA RECORDER | PRICE £29.90 |

### Availability:

These three tape recorders are likely to be sold as compatible with the Electron. The Boots and Smith's machines are readily available through their many retail outlets. The Acorn Data Recorder is available through Acorn dealers or direct from Vector Marketing, Dennington Industrial Estate, Wellingborough, Northants NN8 2RL.

The Smith's and Boots recorders are of a similar shape and overall design, being flat and oblong with six control buttons at the front. The Acorn Data Recorder is slightly different, having its controls on its long edge with the tape inserted beside the controls. All three recorders are of similar average size, about 10 inches long, 6 inches wide and 2 inches high. They all have a flip up lid on top, into which the cassette is inserted.

Other common features include automatic stop/turn off, when reaching the end of a cassette during playback or record, but not during fast forward or rewind. All the machines, except the Data Recorder, can be used in the normal way without the computer, and as such include a built-in microphone and volume control.

All tests on the tape recorders were made using the Electron introductory cassette, supplied by Acorn, tapes produced on the machine itself, and tapes produced on other machines. Most cassette recorders are capable of recording and playback at two different levels, through different sockets and leads. The DIN input/output socket usually has three or five pins. The

jack sockets are 3.5 mm in size and will be in a pair on the recorder, one for input, one for output. These two are accompanied, on the machines reviewed here, by a third and smaller (2.5 mm) socket. This allows motor control from the computer.

### 1. BOOTS CR325

The overall appearance of this machine is very good. It has a tape counter which is easily read, and sliding volume and tone controls, all easily accessible on the top. The controls incorporate a single button for stop/eject and also include a pause button, which can be very useful when using it with the computer.



Both DIN and 3.5 mm jack sockets are available for record/playback. These are located at the front of the machine, below the control buttons. Positioned here they seemed a little in the way, most machines having them to the side.

**PERFORMANCE:** This machine loaded the introductory cassette and its own tapes reliably. However it would not load tapes produced on the Smith's machine.



## 2. W.H.SMITH'S CCR800

This recorder is sold by Smith's to accompany their range of home computers. This tape recorder is very slim and has an impressive appearance. A tape counter is provided and is easily read, being located on the top of the machine.



The volume control is located on the left side of the machine, and is not accompanied by any tone controls. The CCR800 features a 'compute mode' switch, which is recommended for use when saving from the computer. This

fixes the automatic recording level control to an 'optimum' level.

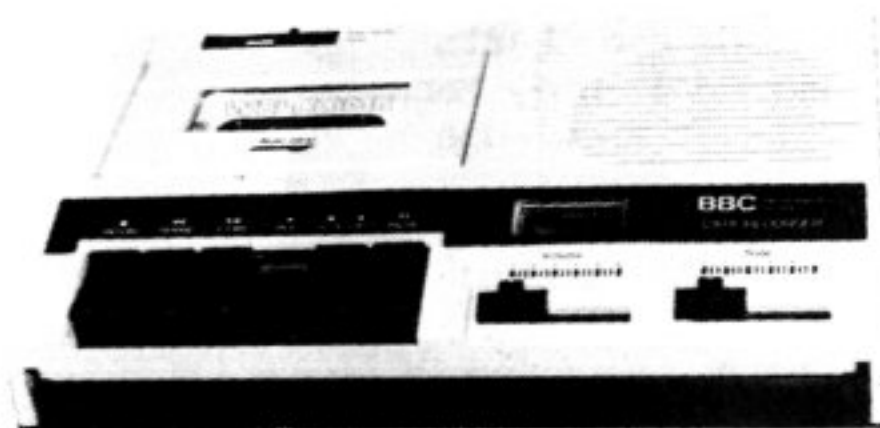
This machine also offers another very useful and unusual feature; the ability to 'cue' and 'review'. This means that if the rewind button is pressed while the play button is locked, the tape will quickly rewind, without resetting the play button, i.e. as soon as the rewind button is released, play immediately continues again. The same applies to the fast-forward control. This really is very handy and allows very quick location of, say, the start of a certain program. Unfortunately the machine does not also offer a pause button.

Both DIN and 3.5 mm jack sockets are provided, being located on the left side of the machine.

**PERFORMANCE:** This machine loaded its own tapes, and those of the Boots machine. It was sometimes difficult to load a program from our introductory cassette.

## 3. ACORN DATA RECORDER

This new recorder replaces the Ferguson 3T07 recorder previously supplied by Acorn. It is not, contrary to rumour, a digital recorder. The name 'Data Recorder' does not appear to signify any radically new design. Externally this recorder is an attractive beige in colour, although this has a tendency to come off very easily. Our Review machines have all scuffed very quickly to reveal the white plastic underneath. It is reasonably well presented with all the controls at the front together with the easily readable VU meter. The tape counter, however, is behind the tape and is rather small and hard to read. All the standard controls are present including pause. It is unfortunately not possible to "cue"; i.e. have play and fast forward down at the same time, a facility which would greatly speed up searching for programs.



Being designed for use with the Electron and the BBC micro, this cassette recorder has only a single 7 pin din socket. The record and playback levels are fixed so that you cannot set them incorrectly. The result of this is that the user has no control whatsoever on the levels used by the recorder. The volume and tone controls only affect the speaker output so that you can listen to your program loading quietly. The machine has motor control,



working through the 7 pin connector. Unfortunately there is no override on this so that you are forced to use the computer's motor control commands to rewind the tape. This can be very inconvenient. Most tape recorders have the motor control on a separate plug which can be easily removed.

**PERFORMANCE:** The data recorder proved much the best recorder of the three during tests, being completely reliable on all the tapes we have tried. However the total lack of control over the recorder's output can lead to problems, as we found when reviewing another sample of this recorder for the BBC micro.

## CONCLUSION

The Boots recorder is probably the best buy for those trying to obtain a reasonably cheap tape recorder, as it would load reliably on most tapes and is a well built and robust recorder.

The Smith's machine was no more reliable in our tests, and is more expensive. However, it is compact and is very attractive visually.

The Data Recorder gave the best results in our tests, though we have had problems with other samples. For those wishing to spend a little less, the Boots recorder was reasonably reliable and survives punishment much better.

## NEWS

### PROGRAMS FROM ACORNSOFT

Acornsoft have announced 12 software packages for release at the same time as the Electron. Acornsoft already have a high reputation for their software for the BBC micro and these versions for the Electron will maintain that standard. This first batch contains games, utilities and two extra languages. The full list is as follows:

- \* Creative Graphics
- \* Graphs and Charts
- \* Personal Money Management
- Tree of Knowledge
- LISP Language
- FORTH Language
- \* Starship Command
- \* Monsters
- \* Chess
- \* Draughts and Reversi
- Snapper
- \* Meteors

\* These programs are reviewed in this issue of ELBUG.

Acornsoft programs are available from Acornsoft Ltd, c/o Vector Marketing Ltd, Dennington Industrial Estate, Wellingborough, Northants NN8 2RL, and from computer shops and dealers.

### ELECTRON TO BE SOLD BY W.H.SMITH'S

As well as specialist computer shops and dealers, the new Electron will also

be sold in some 200 branches of W.H.Smith & Sons Ltd throughout the country. First supplies are expected to have reached the shops by mid October. W.H.Smith are also setting up in-house 'Computer Shops' in 22 of their larger branches.

### ELECTRON ADD-ONS FROM SIR COMPUTERS

SIR Computers of Cardiff are planning to have a number of add-ons available for the Electron by the end of October. These will all plug into the connector at the rear of the micro to provide additional capability. The first of these will provide two extra ports, a Centronics compatible port for connecting a printer to the Electron, and an analogue port, particularly for connecting joysticks to use with games. The second add-on will be an expansion box to accommodate software supplied in ROM format. This will enable the Electron to use the same kind of ROM chips as currently produced for the BBC micro, like the word processing package WORDWISE, and the machine code monitor, EXMON. The third add-on will provide the Electron with a serial RS423 port for communication with other computers.

Further information from SIR Computers Ltd., 91 Whitchurch Road, Cardiff. Telephone (0222) 21341.



# ELECTRON BOOK INDEX

Compiled by Masoud Yazdani

Anyone buying a new Electron will receive with it a copy of the book, "Start Programming with the Electron", by Masoud Yazdani, and published by Addison-Wesley. This otherwise excellent book is marred by the lack of an index. We are most grateful to the author for supplying us with a full index, which, we understand, is to be included in subsequent reprintings of the book.

## A

|                       |     |
|-----------------------|-----|
| Acorn                 | 117 |
| AND                   | 34  |
| argument              | 64  |
| array                 | 70  |
| ASC                   | 66  |
| ASCII codes           | 66  |
| assembly language     | 117 |
| assignment expression | 34  |
| auto-repeat           | 89  |

## B

|              |    |
|--------------|----|
| BASIC        | 3  |
| block number | 20 |
| blocks       | 20 |
| bugs         | 7  |

## C

|                         |     |
|-------------------------|-----|
| CHAIN                   | 83  |
| CHRS                    | 66  |
| CLG                     | 96  |
| CLS                     | 5   |
| conditional branching   | 20  |
| coordinates (relative-) | 103 |
| cursor (edit-)          | 6   |
| cursor (graphic-)       | 97  |

## D

|           |    |
|-----------|----|
| DATA      | 59 |
| data      | 59 |
| debugging | 7  |
| DEF       | 5  |
| DIM       | 70 |
| dimension | 70 |
| DRAW      | 97 |

## E

|          |    |
|----------|----|
| editing  | 4  |
| editor   | 15 |
| Electron | 3  |
| ELSE     | 19 |
| END      | 8  |
| ENDPROC  | 5  |
| ENVELOPE | 87 |
| ERL      | 91 |

|                   |     |
|-------------------|-----|
| error messages    | 7   |
| error-trapping    | 91  |
| Exeter University | 117 |
| expression        | 32  |

## F

|                      |     |
|----------------------|-----|
| FALSE                | 34  |
| FOR                  | 18  |
| FORTH                | 117 |
| functions (inbuilt-) | 35  |
| functions (own-)     | 35  |

## G

|                 |     |
|-----------------|-----|
| GCOL            | 105 |
| GOTO            | 91  |
| GREETER program | 82  |

## I

|                       |    |
|-----------------------|----|
| IF                    | 19 |
| immediate mode        | 4  |
| information           | 58 |
| INKEY                 | 90 |
| INPUT                 | 61 |
| INSTR                 | 78 |
| introductory cassette | 24 |

## K

|                  |   |
|------------------|---|
| keyboard         | 1 |
| keyword (BASIC-) | 4 |

## L

|        |    |
|--------|----|
| LEFT   | 18 |
| LEFT\$ | 78 |
| LEN    | 75 |
| LIST   | 5  |
| LOAD   | 20 |
| LOCAL  | 36 |
| LOGO   | 30 |
| loop   | 18 |

## M

|                 |    |
|-----------------|----|
| machine code    | 14 |
| mishap messages | 7  |
| MODE            | 94 |



|                        |     |
|------------------------|-----|
| modes                  | 93  |
| MOVE                   | 97  |
| <b>N</b>               |     |
| name                   | 18  |
| NEW                    | 12  |
| NEXT                   | 18  |
| NOT                    | 35  |
| null character         | 48  |
| <b>O</b>               |     |
| OLD                    | 37  |
| ON ERROR               | 91  |
| Operating System       | 15  |
| OR                     | 34  |
| <b>P</b>               |     |
| Papert, S.             | 30  |
| pitch                  | 85  |
| pitch increment        | 88  |
| pixel                  | 96  |
| PLAY                   | 20  |
| PLOT                   | 98  |
| PRINT                  | 4   |
| PRINTTAB               | 22  |
| PROC                   | 5   |
| procedure              | 5   |
| program (computer-)    | 1   |
| PROLOG                 | 117 |
| prompt                 | 3   |
| <b>R</b>               |     |
| READ                   | 59  |
| RECORD                 | 20  |
| recursion              | 12  |
| REM                    | 8   |
| REPEAT                 | 34  |
| reports                | 7   |
| resolution             | 95  |
| RESTORE                | 60  |
| RIGHT\$                | 84  |
| RND                    | 35  |
| RUN                    | 9   |
| <b>S</b>               |     |
| SAVE                   | 20  |
| SOLVER                 | 46  |
| SOUND                  | 85  |
| SPC                    | 64  |
| SQR                    | 35  |
| STEP                   | 18  |
| step number            | 88  |
| strings                | 67  |
| structured programming | 39  |
| subscript              | 70  |

**T**

|                     |    |
|---------------------|----|
| TAB                 | 63 |
| THEN                | 19 |
| TO                  | 18 |
| top-down refinement | 39 |
| TRUE                | 34 |
| TURTLE              | 24 |
| turtle graphics     | 30 |
| TV                  | 3  |

**U**

|            |    |
|------------|----|
| UNTIL      | 34 |
| User Guide | 3  |

**V**

|          |    |
|----------|----|
| value    | 18 |
| variable | 18 |
| VDU      | 95 |
| voices   | 85 |

## Special Characters:

|                         |    |
|-------------------------|----|
| "                       | 4  |
| \$                      | 68 |
| ↓                       | 6  |
| "2                      | 4  |
| →                       | 6  |
| ←                       | 6  |
| ?/                      | 4  |
| BREAK                   | 3  |
| CAPS LK                 | 3  |
| COPY                    | 6  |
| DELETE                  | 4  |
| ESCAPE                  | 12 |
| FUNC                    | 4  |
| G                       | 21 |
| R                       | 21 |
| RETURN                  | 3  |
| SHIFT                   | 3  |
| I                       | 6  |
| *                       | 32 |
| *FX                     | 91 |
| +                       | 33 |
| -                       | 32 |
| /                       | 33 |
| <                       | 34 |
| <>                      | 34 |
| = (function result)     | 36 |
| = (logical equivalence) | 38 |
| = (with LET)            | 18 |
| >                       | 34 |
| ^                       | 37 |
| :                       | 91 |

# HEDGEHOG

by Alan Dickinson

Hedgehog is a really first class implementation of the "Frogger" type arcade game. It is fast moving, responsive, the graphics are good, and it makes pretty addictive playing. Essentially you are a hedgehog. You can move left, right or "scurry". Your aim in life is to scurry across a four lane dual carriageway and a busy railway track to gather acorns for supper. It is the rush hour, so the traffic is pretty busy, which means you need concentration and good "scurry" control. If you do get run over by a juggernaut or an Intercity 125, an ambulance will quickly drive to the scene.

You have three lives, and your score depends on your progress. As your score increases, so the traffic gets steadily worse. Playing tips:

1. Do not roll into a ball in the face of oncoming traffic.
2. Make good use of the rest point between the road and the railway track.

If you find the game too fast, then you may like to modify it to run in mode 2. It is a 'feature' of the Electron that it runs much slower in this mode. The mode is set in line 90. A further advantage of mode 2 is that you can use more colours in the display. To achieve this, change line 1810 to

```
1810 COLOUR (R&MOD5)+2
```



```
10 REM PROGRAM. HEDGEHOG
20 REM AUTHOR. Alan Dickinson
30 REM VERSION E1.2
40 REM ELBUG November 1983
50 REM Subject to COPYRIGHT
60 :
70 ON ERROR ON ERROR OFF:MODE6:REPOR
T:PRINT " at line ";ERL:END
80 :
90 MODE5
```

```
100 VDU23,1,0;0;0;0;
110 *FX9,25
120 *FX10,25
130 DIM R$(9):REM ROADS
140 DIM A$(19):REM ACORNS
150 :
160 PROCINTRO
170 PROCDEFINE
180 PROCINIT
190 REPEAT
200 PROCGAME
210 IF S%>T% PROCTOPSCORE
220 PROCGAMEOVER
230 TIME=0:REPEAT UNTIL TIME>200
240 *FX15,1
250 A$=GET$
260 UNTIL FALSE
270 :
280 DEF PROCINTRO
290 COLOUR 133
300 CLS
310 COLOUR 15
320 PRINT"" Hedgehog"
330 COLOUR 4
340 PRINTTAB(2,14)"Controls..."
350 PRINT"" Z = LEFT"
360 PRINT"" X = RIGHT"
370 PRINT"" / = SCURRY"
380 PRINTTAB(2,30)"Snatch acorns..."
;
390 TIME=0:REPEAT UNTIL TIME>300
400 COLOUR 8
410 PRINTTAB(2,30)"Press any key..."
;
420 A$=GET$
430 ENDPROC
440 :
450 DEF PROCDEFINE
460 VDU23,255,136,204,238,238,238,23
8,238,238
470 VDU23,254,0,0,-1,-1,0,0,0,0
480 VDU23,252,0,-1,-1,0,0,-1,-1,0
490 VDU23,251,0,0,0,0,-1,-1,0,0
500 VDU23,253,24,60,90,255,126,255,6
0,0
510 VDU23,250,108,72,200,252,254,251
,127,60
520 VDU23,249,24,60,60,60,0,126,60,24
530 H$=CHR$17+CHR$1+CHR$253
540 VDU23,224,15,105,105,249,255,255
,255,102
550 VDU23,225,240,150,150,159,255,25
5,255,102
560 VDU23,226,0,0,0,126,126,126,255,
102
```





```

570 VDU23,227,0,126,126,126,126,126,
255,102
580 VDU23,228,31,49,97,255,255,255,2
55,48
590 VDU23,229,255,36,36,255,255,255,
255,0
600 VDU23,230,248,140,134,255,255,25
5,255,24
610 VDU23,231,0,31,17,17,255,191,255
,48
620 VDU23,232,0,240,136,136,255,253,
255,12
630 VDU23,233,255,255,255,255,255,25
5,255,28
640 VDU23,234,255,239,199,239,255,25
5,221,28
650 ENDPROC
660 :
670 DEF PROCINIT
680 T%=500
690 T$="Henry Hedgehog"
700 S1$=" ":S2$=" "
710 S3$=" ":S4$=" "
720 S5$=" ":S6$=" "
730 R$(0)=STRING$(20," ")
740 R$(1)=S2$+CHR$227+CHR$226+CHR$22
7+CHR$227+CHR$225+S4$+CHR$226+CHR$225+S
5$+CHR$227+CHR$226+CHR$225+S4$
750 R$(2)=CHR$224+CHR$226+CHR$227+CH
R$226+CHR$226+S5$+CHR$224+CHR$226+S3$+C
HR$224+CHR$227+CHR$227+CHR$226+S2$+CHR$
224+CHR$226+S3$
760 R$(3)=CHR$228+CHR$229+CHR$229+CH
R$230+S4$+CHR$227+CHR$227+CHR$225+S6$+C
HR$228+CHR$229+CHR$230+S3$+CHR$226+CHR$
225+S1$
770 R$(4)=CHR$228+CHR$229+CHR$229+CH
R$230+S4$+CHR$224+CHR$227+CHR$227+S6$+C
HR$228+CHR$229+CHR$230+S3$+CHR$224+CHR$
226+S1$
780 R$(5)=STRING$(20," ")
790 R$(6)=S2$+CHR$228+CHR$229+CHR$22
9+CHR$232+S5$+CHR$231+CHR$232+S5$+CHR$2
33+CHR$233+CHR$232+S1$+CHR$231+CHR$232+
S2$
800 R$(7)=CHR$231+CHR$232+S2$+CHR$23
1+CHR$232+S2$+CHR$233+CHR$232+S5$+CHR$2
28+CHR$232+S1$+CHR$233+CHR$232+S5$
810 R$(8)=CHR$231+CHR$232+S1$+CHR$23
1+CHR$232+S3$+CHR$231+CHR$233+S5$+CHR$2
31+CHR$230+S1$+CHR$231+CHR$233+S5$
820 R$(9)=S4$+CHR$231+CHR$232+S1$+CH
R$231+CHR$229+CHR$229+CHR$230+S3$+CHR$2
31+CHR$233+S3$+CHR$231+CHR$233+S3$+CHR$
231+CHR$230+S1$
830 FOR I%=1 TO 9
840 R$(I%)=STRING$(5,R$(I%))
850 R$(I%)=LEFT$(R$(I%),100)
860 NEXT
870 ENDPROC

```



```

880 :
890 DEF PROCGAME
900 S%=-100:LEVEL%=-1:PROCHARDER
910 N%=0
920 REPEAT
930 N%=N%+1:PROCGO
940 UNTIL N%=3
950 IF S%<1000 ENDPROC
960 COLOUR 0:COLOUR 129:CLS
970 PRINTTAB(2,10)"Bonus hedgehog";
980 PRINTTAB(2,20)"May the fleas";
990 PRINTTAB(2,22)" be with you";
1000 FOR I%=0 TO 255 STEP8
1010 SOUND&1,-12,I%,1
1020 NEXT
1030 *FX15,1
1040 A$=INKEY$(200)
1050 REPEAT:N%=N%+1:PROC_GO:UNTIL N%=
4
1060 ENDPROC
1070 :
1080 DEF PROCGO
1090 X%=10:Y%=30:Z%=10
1100 PROCDRAWSCREEN
1110 REPEAT
1120 PROCROAD(1,0,12)
1130 PROCROAD(6,0,22)
1140 PROCROAD(7,0,24)
1150 PROCROAD(4,1,18)
1160 PROCROAD(8,1,26)
1170 PROCROAD(9,1,28)
1180 PROCROAD(7,0,24)
1190 PROCROAD(2,1,14)
1200 PROCROAD(8,1,26)
1210 PROCROAD(3,0,16)
1220 UNTIL Y%=0
1230 IF ACORNS%=0 PROCHARDER
1240 ENDPROC
1250 :
1260 DEF PROCDRAWSCREEN
1270 COLOUR 0
1280 COLOUR 128
1290 CLS
1300 COLOUR 129
1310 PRINTTAB(1,4)SPC(17);
1320 PRINTTAB(1,5)" H E D G E H O G "
;
1330 PRINTTAB(1,6)SPC(17);
1340 COLOUR3
1350 COLOUR 128
1360 PRINTTAB(1,1);N%;
1370 PRINTTAB(0,10)STRING$(20,CHR$255
)
1380 COLOUR 9
1390 FOR I%=0 TO 19
1400 IF A$(I%)=1 PRINTTAB(I%,10)CHR$2
49
1410 NEXT
1420 COLOUR 7
1430 PRINTTAB(0,21)STRING$(20,CHR$254)

```



```

1440 PRINTTAB(0,23)STRING$(10,"- ")
1450 PRINTTAB(0,25)STRING$(20,CHR$252
)
1460 PRINTTAB(0,27)STRING$(10,"- ");
1470 PRINTTAB(0,29)STRING$(20,CHR$251
)
1480 PRINTTAB(10,1)"Score ";S%;
1490 COLOUR13
1500 PRINTTAB(0,20)R$(5);
1510 PRINTTAB(X%,Y%)H$;
1520 ENDPROC
1530 :
1540 DEF PROCHOG
1550 IF Y%=0 ENDPROC
1560 IF INKEY-98 AND X%>0 PROCLEFT EL
SE IF INKEY-67 AND X%<19 PROCRIGHT ELSE
IF INKEY-105 PROCFWD
1570 IFZ%<10 IF MID$(R$(Z%),X%+1,1)<>
S1$ PROCSPLAT
1580 ENDPROC
1590 :
1600 DEF PROCLEFT
1610 X%=X%-1:PRINTTAB(X%,Y%)H$;" ";
1620 ENDPROC
1630 :
1640 DEF PROCRIGHT
1650 PRINTTAB(X%,Y%) " ";H$;:X%=X%+1
1660 ENDPROC
1670 :
1680 DEF PROCFWD
1690 IF Z%=1 AND A%(X%)<>1 SOUND 1,-1
5,40,1:ENDPROC
1700 Y%=Y%-2
1710 Z%=Z%-1
1720 PRINTTAB(X%,Y%)H$;TAB(X%,Y%+2)S1
$;
1730 SOUND 1,-12,220,1
1740 S%=S%+10
1750 PRINTTAB(10,1)"Score ";S%;
1760 IF Z%=0 PROCCHUCKLE
1770 ENDPROC
1780 :
1790 DEF PROCROAD(R%,D%,SY%)
1800 IF D%>0 R$(R%)=MID$(R$(R%),2,99)
+LEFT$(R$(R%),1) ELSE R$(R%)=MID$(R$(R%
),100,1)+LEFT$(R$(R%),99)
1810 COLOUR (R%MOD2)+2
1820 COLOUR 128
1830 PRINTTAB(0,SY%)LEFT$(R$(R%),20);
1840 IF SY%=Y% PRINTTAB(X%,Y%)H$;
1850 PROCHOG
1860 IF Y%=SY% IF MID$(R$(R%),X%+1,1)
<>S1$ PROCSPLAT
1870 IF RND(80)=80 SOUND3,-15,1,10:SO
UND2,-15,9,8:SOUND1,-15,17,2
1880 ENDPROC
1890 :
1900 DEF PROCSPLAT
1910 IF Y%=0 ENDPROC
1920 COLOUR 14

```



```

1930 PRINTTAB(X%,Y%) "*";
1940 VDU19,0,1,0,0,0
1950 COLOUR 12
1960 COLOUR 139
1970 PRINTTAB(1,4)SPC(17);
1980 PRINTTAB(1,5)" S P L A T "
;
1990 PRINTTAB(1,6)SPC(17);
2000 COLOUR 128
2010 FOR I%=1 TO 15
2020 SOUND 1,-5,255-10*I%,1
2030 NEXT
2040 COLOUR 1
2050 PRINTTAB(X%,Y%)CHR$250;
2060 VDU20
2070 IF X%>10 AX%=0:AMB$=" "+CHR$234+
CHR$232 ELSE AX%=17:AMB$=CHR$231+CHR$23
4+" "
2080 FOR I%=0 TO 15
2090 IF AX%<X% PRINTTAB(AX%,Y%)AMB$;:
AX%=AX%+1
2100 IF AX%>X% AX%=AX%-1:PRINTTAB(AX%
,Y%)AMB$;
2110 FOR J%=0 TO 120 STEP2
2120 SOUND&11,-I%,J%,5
2130 NEXT
2140 NEXT
2150 Y%=0
2160 ENDPROC
2170 :
2180 DEF PROCCHUCKLE
2190 FOR I%=1 TO 25
2200 J%=1
2210 SOUND J%,-15+I%/3,200-RND(3)*I%
,1
2220 NEXT
2230 N%=N%-1:Y%=0:S%=S%+25:A%(X%)=0:A
CORN$=ACORN$-1
2240 ENDPROC
2250 :
2260 DEF PROCHARDER
2270 FOR I%=1 TO 3
2280 FOR J%=30 TO 180 STEP8
2290 SOUND &12,-11-I%,J%+12+I%*10,1
2300 NEXT
2310 NEXT
2320 S%=S%+100
2330 FOR I%=0 TO 19:A%(I%)=0:NEXT
2340 FOR I%=1 TO 4
2350 REPEAT
2360 J%=RND(17)+1
2370 UNTIL A%(J%)=0
2380 A%(J%)=1
2390 NEXT
2400 ACORN$=4
2410 LEVEL%=LEVEL%-2*(LEVEL%<>9)
2420 R$(5)=STRING$(LEVEL%,CHR$255)+ST
RING$(20-LEVEL%*2,"")+STRING$(LEVEL%,C
HR$255)
2430 ENDPROC

```





```

2440 :
2450 DEF PROCTOPSCORE
2460 COLOUR 3
2470 COLOUR 129
2480 CLS
2490 PRINTTAB(1,3)"T O P - S C O R E";
2500 PRINTTAB(1,8)"Enter your name";
2510 PRINTTAB(1,10);
2520 FOR I%=100 TO 200 STEP 12
2530   SOUND 1,-15,I%,3
2540   SOUND 2,-14,I%+8,3
2550 NEXT
2560 *FX15,1
2570 INPUT ""T$
2580 T%=S%
2590 ENDPROC
2600 :
2610 DEF PROCGAMEOVER
2620 COLOUR 4
2630 COLOUR 133

```



```

2640 CLS
2650 PRINTTAB(5,3)"Super hog";
2660 COLOUR 15
2670 PRINTTAB((20-LEN(T$))DIV2,6)T$;
2680 COLOUR 4
2690 PRINTTAB(5,11)"Top score";
2700 COLOUR 0
2710 PRINTTAB(8,13);T%;
2720 COLOUR 4
2730 PRINTTAB(5,18)"Your score";
2740 COLOUR 0
2750 PRINTTAB(8,20);S%;
2760 COLOUR 4
2770 PRINTTAB(4,30)"Press any key";
2780 ENDPROC

```

This program was adapted from a version for the BBC micro first published in BEEBUG Vol.2 No.2.

## HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS HINTS

### TRACE

This is a useful debugging function, allowing you to follow the computer's execution of your program. When it executes a line, the computer prints the line number on the screen in square brackets. From this, you can see the order in which the instructions are executed.

Tracing is initiated by typing:

TRACE ON <return>

and running the program. At the end, turn it off with:

TRACE OFF <return>

You may also use TRACE xxx, where xxx is the line you want the computer to trace up to. After that TRACE is automatically turned off, and normal execution continues.

### SELF VALIDATING GET ROUTINE - M.Girling

When your program needs to ask a question seeking a single key response, it is very messy to check for every possible response in turn with a separate IF statement. The ON...INSTR...GOTO construction is much neater:

```

1000 PRINT "ANOTHER GAME (Y/N)";
1010 ON INSTR("YNyn",GET$) GOTO 100,200,100,200 ELSE 1010

```

This will go to line 100 on a 'yes' response and to line 200 for a 'no' response, and will loop around for another key if an illegal one is typed.

### CALCULATING X SQUARED

When squaring or cubing a number, often needed in games programs, use X\*X (or similar) not X^2 as X\*X is much faster. Incidentally when using the Pythagoras theorem to calculate distances between, say, two spaceships on the screen, don't bother taking the square root, simply adjust the minimum distance to be in units squared.

### PROGRAM LENGTH - Matthew Rapier

To find the length of the program in memory type:

PRINT TOP-PAGE <return>

This prints out the length of the program in bytes. Also, note that there is an abbreviation for PRINT, which is 'P.'



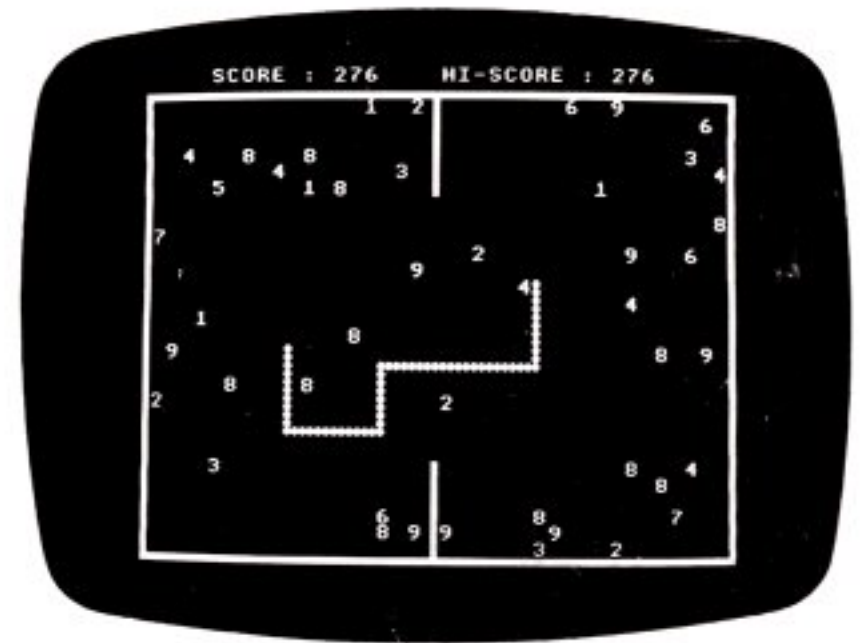
## BEEBUGSOFT

### SNAKE for the Electron

Snake is a very addictive, fast moving arcade game. The purpose of the game is to direct the snake around the screen, eating up the randomly placed objects. With every bite that it takes, the tail grows longer and the snake moves faster. Just to complicate things, there are also certain scattered obstacles which must be avoided.

To even complete the first screen requires skill and concentration. Further frames use different shaped playing areas, which as well as adding variety, require much greater skill to complete.

This game is much more exciting than many games of a similar nature, and once played, proves to be surprisingly compulsive.



Snake costs £4.00 inc VAT. Please add 50p post & packing.  
Overseas orders: £5.50 inc post & packing\-\VAT not charged.

Send order with membership number to:  
BEEBUGSOFT, P.O. Box 109, High Wycombe, Bucks HP11 2TD.





IF YOU WRITE TO US

BACK ISSUES (Members only)

All back issues will be kept in print (from November 1983). Send 90p per issue PLUS an A5 SAE to the subscriptions address. Back copies of BEEBUG are available to ORBIT members at this same price. This offer is for members only, so it is ESSENTIAL to quote your membership number with your order. Please note that the advertising supplements are not supplied with back issues.

Subscription and Software Address

ELBUG  
PO BOX 109  
High Wycombe  
Bucks  
HP11 2TD

SUBSCRIPTIONS

Send all applications for membership, and subscription queries to the subscriptions address.

MEMBERSHIP COSTS:

£5.90 for 6 months (5 issues)  
£9.90 for 1 year (10 issues)  
European Membership £16 for 1 year.  
Elsewhere (Postal zones)  
Zone A £19, Zone B £21, Zone C £23

SOFTWARE (Members only)

This is available from the software address.

IDEAS, HINTS & TIPS, PROGRAMS  
AND LONGER ARTICLES

Substantial articles are particularly welcome and we will pay around £25 per page for these, but in this case please give us warning of anything that you intend to write.

We will also pay £10 for the best Hint or Tip that we publish, and £5 to the next best. Please send all editorial material to the editorial address opposite. If you require a reply it is essential to quote your membership number and enclose an SAE.

Editorial Address

ELBUG  
PO Box 50  
St Albans  
Herts  
AL1 2AR

ELBUG MAGAZINE is produced by BEEBUG Publications Ltd.

Editor: Mike Williams. Technical Editor: Philip Le Grand.

Production Editor: Phyllida Vanstone. Technical Assistant: Alan Webster.

Managing Editor: Lee Calcraft.

Thanks are due to David Graham, Sheridan Williams, Adrian Calcraft and Matthew Rapiere for assistance with this issue.

All rights reserved. No part of this publication may be reproduced without prior written permission of the Publisher. The Publisher cannot accept any responsibility, whatsoever, for errors in articles, programs, or advertisements published. The opinions expressed on the pages of this journal are those of the authors and do not necessarily represent those of the Publisher, BEEBUG Publications Limited.  
BEEBUG Publications LTD (c) November 1983.